uc3m | Universidad Carlos III de Madrid

University Degree in Energy Engineering
2018-2019

Bachelor Thesis

"Sarcasm recognition survey and application based on Reddit comments"

Nicolás Hernando Alcalá

Tutor

José Antonio Iglesias Martínez

Leganés, July 2019

ABSTRACT

Social media platforms are continuously increasing their number of users, and every day enormous amounts of data are produced online. Machine Learning (ML) techniques in the form of speech recognition are applied to analyze the polarity of this unstructured text data.

However, it is broadly used sarcasm through these platforms, reducing the accuracy of said systems, as the intention of the message expressed does not match the polarity that is measured.

Throughout the development of this work a survey considering three different algorithms will be performed. These algorithms are Logistic Regression, Neural Networks and Support Vector Machines.

This final degree project proposes a previous analysis to the data using a sarcasm recognition classifier implemented with a support vector machine algorithm, with a mean accuracy of 71.21% and an F1-Score around 60%.

Finally, an analysis of the planification and the costs is performed, proposing future works that could complement this bachelor thesis.

Key Words: Machine Learning (ML); Supervised Learning; Support Vector Machines (SVM); Speech Recognition; Sarcasm Recognition.

INDEX

FIGURES INDEX

TABLE INDEX

1. INTRODUCTION

In this chapter it will be covered the contents of the introduction of this bachelor thesis.

1.1. Motivation

The motivation of this final degree project is driven by different aspects. First, try to construct a simpler sarcasm detection system, that could be used easily, as it is a major problem of the speech recognition techniques existing at the moment. For instance, to know the public opinion about the electric bill or general politic positioning about a certain topic.

Moreover, to develop a survey regarding some of the different technologies available for such purpose, from a mathematical perspective, in order to make it easier for others to get introduced in the problem descripted.

Finally, to develop an entire functioning Machine Learning algorithm using a free programming language and platform, trying to make the process more accessible, via Octave with small contributions in MATLAB..

1.2. Problem Description

The amount of information generated every day by the society is enormous. Forbes [1] expressed that the data generated over 2017 and 2018 represented by itself 90 percent of the data that has been generated in the world. This is done by means of images, videos, news and social media.

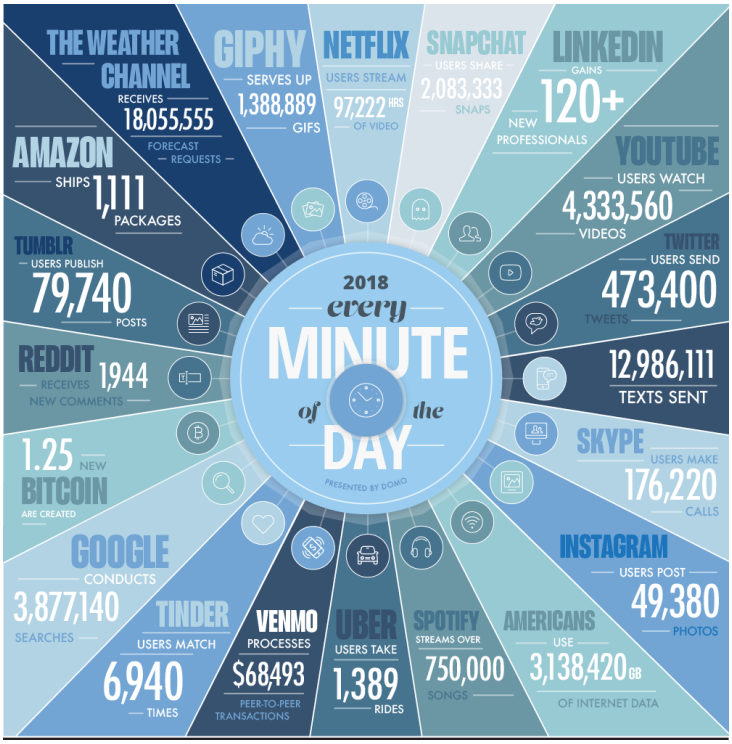

Fig. 1.1 Data generated every minute [2].

Social media has been experimenting a continuous enlargement that does not seem to stop. It's a growing industry that generates billions of dollars every year. According to Domo, data never sleeps report [2], it is estimated that in 2020 1.7MB of data will be generated every second.

Furthermore, the international data corporation, developed a report [3] in 2016 stating that 90% of the internet data is unstructured, mostly in the form of text. This brings up the problem regarding what to do with all this data. It is of general interest, from businesses to other actors, an analysis that explains the contents of that information.

Speech recognition is one of the most relevant mechanisms that can help along with this problem. By means of the usage of this technology, the polarity of a text or comment can be determined with a high grade of success. However, some of the most problematic issues regarding this tool have to do with sarcasm recognition. This happens because the polarity expressed does not match the hidden intention of the comment.

In everyday conversations sarcasm can be present and with the profound usage of social media it can be encountered a massive use of this linguistic form of speech. Therefore, it is necessary to find general ways of performing speech recognition with the addition of irony detection.

1.3. Objective

The objectives that this bachelor thesis pursues can be compressed in the three more relevant:

- Study and develop a survey on Machine Learning algorithms that can be used for speech recognition, and more concretely, for sarcasm detection.

- Create an effective algorithm for said field, choosing between one of the studied along the survey and basing the choice in the algorithms previously used in investigations regarding irony detection.

- Try to simplify the problem obtaining the highest performance possible considering these simplifications. By means of this, the objective is to make the algorithm more accessible.

1.4. Regulatory Framework

The regulatory framework will be analyzed along this section. The laws that are to be considered in this bachelor thesis will be from Spain. This election was made following that the work has been developed in this location.

Intellectual protection law (LPI R.D. 1/1996 at the BOE number 97, 22 April 1996) determines that the intellectual author of a work has all the rights reserved amongst it.

The author can allow its usage with no conditions, allow its usage under reference citation, or not allow it at all. Moreover, author can allow or deny its usage under certain conditions of publication, either if public or private. This apply to the images along the work and papers.

Article 32.1 allows the usage of others work in case that an opinion or judgement is made upon it, always by referencing it. If the work developed is to be presented with the intention of earning money, the authors must be contacted. This justifies the use of the images and pieces of code for the development of the thesis.

Analyzing the dataset and the code obtained from other sources, which are, the Porter Stemmer and other minor pieces of code along the algorithm implementation, they are under MIT and Apache license. Both of these licenses are not restrictive in neither the academic purpose nor economic.

Finally, it was studied the possibility that the General Data Protection Regulation (LPDP. 3/2018 at the BOE number 294, 6 December 2018) may affect the dataset contents. However, it does not apply to it as the only information saved along the comments that referred to the emitter was the username.

1.5. Operating Environment

In this section, both hardware and software tools used along the development of this final degree project will be presented.

Hardware tools were mostly the personal computer, which characteristics are:

- Computer processing unit: Intel® Core™ i5-8250U CPU @1.60GHz 1.80GHz.
- RAM installed: 8.00GB (7.89GB usable).
- Windows OS 10 Home 64 bits.

The software that is being used is:

- Microsoft Office Tools:
    - Microsoft Word 2018.
    - Microsoft Excel 2018.
    - Microsoft PowerPoint 2018.
- As a text editor, both MATLAB and WordPad.
- Platform for Machine Learning implementation: Octave and MATLAB.

1.6. Socioeconomic Impact

Socioeconomic impact of the project will be evaluated in the project management chapter 6.3, along with small contributions in the state of art chapter, in the sections 2.1, 2.2, 2.4.

Not only contributions to society will be explained, but also the possibilities of this technique to generate income.

1.7. Acronyms and Abbreviations

AI: Artificial Intelligence.

BOB: Bag-Of-Bigrams.

BOW: Bag-Of-Words.

cosSim: Cosine Similarity.

BFGS: Broyden-Fletcher-Goldfarb-Shanno.

CS: Computer Science.

L-BFGS: Limited-BFGS.

LM: Language Model.

LogR: Logistic Regression.

ME: Maximum Entropy.

ML: Machine Learning.

MLE: Maximum Likelihood Estimation.

MSE: Mean-Squared-Error.

NB: Naïve Bayes.

NLP: Natural Language Processing.

POS: Parts of Speech.

SMO: Sequential Minimal Optimization.

SVM: Support Vector Machine.

1.8. Definitions

LIWC*_P: Presence of dictionary-based lexical and pragmatic factors.

LIWC*_F: Frequency of dictionary-based lexical and pragmatic factors.

Stem: Group of words with the same “root”.

Survey Paper: Comprised interpretation of a broad number of papers along a topic with a conclusion.

2. STATE OF THE ART

In this chapter it will be presented the state of the art of the different concepts covered along the thesis. It will also be included the previous work in sarcasm detection using Machine Learning techniques, and the basis of the Support Vector Machine algorithm and the N-gram analysis.

Machine Learning will be separated into different topics in order to explain briefly the notion later applied. When explaining ML, most of the examples used to explain the concepts will refer to linear regression, as it is the basic algorithm and simplifies the reasoning.

First, the notion of sarcasm and its usage in society will be presented. Then, Machine Learning will be briefly explained as well as an introduction along the algorithms that put together form the survey of the bachelor thesis. Moreover, Natural Language Processing and N-gram models will be introduced.

2.1. Sarcasm in Society

Sarcasm is defined as "the use of irony to mock or convey contempt" [4] by the Oxford English Dictionary. It is the use of nonliteral language, where the speaker uses a polarity along the sentence with the intention of transmitting the opposite idea. Along the development of this thesis, sarcasm and irony concepts will be used interchangeably as there is not a clear distinction between both terms.

Sarcasm usually relies on the expression of irony with the purpose of humor. There can be distinguished two types of sarcasm: verbal and written. Verbal sarcasms lean on the combination of different intonation and facial gestures that support the mocking hidden under the sentence that is being expressed. Whereas written sarcasm is difficult to recognize as it does not have the aforementioned characteristics of the verbal sarcasm and only relies on the contradiction of sentiments conveyed in the comment.

Haiman considers in his book [5], based on the premise that language defines humans, that sarcasm is a higher form of expression, and somehow it requires more intelligence. He also reflects the idea that a sarcastic comment has no intention to lie, the speaker wants the recipient to understand the hidden message. He sums up sarcasm as a sentence with a metamessage of "I do not mean this".

It can be observed, with the appearance of social media, what may seem as an increase of the irony usage in society. Its use in the interactions among the users of these platforms has rocketed in the past years, even with the introduction of *hashtags* to make the understanding of the underlying concept easier to the receiver of the message. By means of those markers it is simple to recognize the use of sarcasm in a comment, though they are not that common.

2.2. Machine Learning

The analysis that is to be developed through this section is a combination of different sources, including an online course specialized on Machine Learning. The notation chosen to express the equations will be from Stanford teacher Andrew Ng. ML course [6].

Tom M. Mitchell defines ML as the process where a "Machine learns with respect to a particular task T, performance metric P, and type of experience E, if the system reliability improves its performance P at task T, following experience E" [7]. The definitions of T, P and E may depend on the learning task that the machine is performing. In other words, ML is the process where a machine learns from labeled or unlabeled data through an implementation step and gives results or insights based on that initial data.

There are significant differences between ML and computer science (CS). CS requires to manually program computers, it is mostly based on the statistics that can be obtained through an implementation of a written code. Whereas in ML, the outcome of the program is based on the result of an insight obtained from the data that feeds the program. It is expected that the program learns by itself what are the sensible outcomes of the data.

As explained by Tom M. Mitchell, it is not fully understood how the learning process of animals and, more precisely humans, works. However, the junction created between computers and humans has proven to be positive, as it increases the capacity to cross the settled limits of computing and processing information, enlarging the knowledge of the surrounding universe, for instance.

Nowadays, ML is useful in a broad branch of applications. From computer vision, working with sciences based on empirical results, recommender systems, such as Netflix and Amazon, to speech recognition. ML is present in lots of the day to day commodities that are taken from granted in recent years.

When determining the steps to follow for choosing the correct algorithm, Pedro Domingos [8] expresses the learning process as a combination of representation, evaluation and optimization. Firstly, the programming language and environment should be selected, hypothesis made, and the features chosen. Later, an evaluation along the performance of the algorithms should be assessed. Finally, choosing the highest score among the algorithms and select the learning process that best suits the desired result is required.

ML learning processes are divided into two categories: supervised and unsupervised learning.

2.2.1. Supervised Learning

In supervised learning, the data from where the algorithm obtains its information is labeled. Therefore, the output of the training examples is known. The ML system gets the information of the output of each example, taking into account the value of the features, or characteristics, that are present in each example to create a hypothesis that will determine the answers of new unlabeled information.

There are two main types of supervised learning depending on the output:

- Regression: The algorithm predicts a continuous value for the output. For instance, the price of a house given certain features.
- Classification: The algorithm predicts a discrete value for the output. For instance, if a student will pass or not a certain exam, 0 or 1 case.

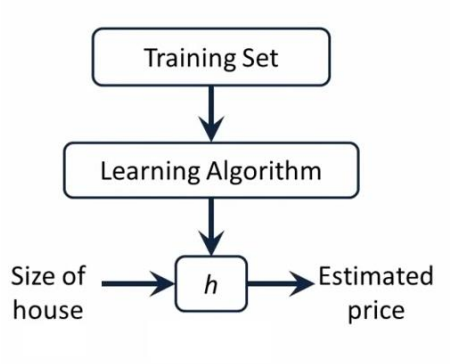

Fig. 2.1 ML development [6].

2.2.2. Unsupervised Learning

In unsupervised learning the algorithm is supposed to find structure in unlabeled data. It is also called clustering algorithm, as it should find different clusters in data, commonly using centroids to solve the minimum distance from the data it uses.

This type of learning process in ML is useful to approach problem where little is known, an begin to have some insights that could lead to new developments on the research and interesting discoveries.

2.2.3.Hypothesis and Cost Function

First, it is important to determine the notation that will be used from now on along the evolution of this thesis. The letter m is the number of examples in the training set, $x's$ will refer to the feature values or X for the feature matrix, whereas y will represent the output of the equation. The hypothesis equation will be expressed by h_θ or h interchangeably, where θ is the variable which value will be optimized. A single value of a set will be determined by (x, y) and $(x^{(i)}, y^{(i)})$ will represent the i^{th} training example. Training examples are the ones from which the algorithm learns. As referred at the beginning of this section, the notation of the equations will be the one used in Andrew Ng. ML Coursera course [6].

In supervised learning, the hypothesis points, should match the values of the output variable. Therefore, $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_n)$, should be optimized so that the difference between the hypothesis and the output is minimum.

To measure this distance, the cost function $J(\theta)$ is used. The goal in the learning process is to minimize this function, which can be considered as the error between h_θ and y.The simplest version of the hypothesis and cost function can be found in linear regression with one variable:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 \tag{2.1}$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2 \tag{2.2}$$

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1) \tag{2.3}$$

Having x_0 as a bias unit used to generalize the expression:

$$h_\theta = g(\theta^T X) \tag{2.4}$$

2.2.4.Gradient Descent

It is important to explain the basic implementation process that occur in many ML algorithms, such as, linear regression. The minimization arises from a continuous operation of trying different theta values, in order to reduce the cost function value, until convergence, ideally in the global minimum. The first value approximation determines the number of iterations and the minimum that would be reached, as it can be seen in figure 2.2. For linear regression it would be:

$$\theta_0 = \theta_0 - \alpha \frac{\delta}{\delta\theta_0} J(\theta) \tag{2.5}$$

$$\theta_j = \theta_j - \alpha \frac{\delta}{\delta\theta_0} J(\theta) \tag{2.6}$$

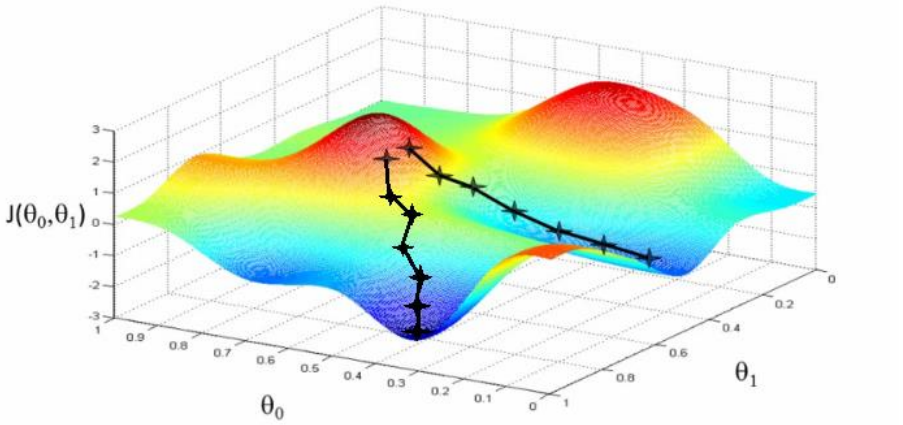

Fig. 2.2 Gradient descent [6].

Where α is the learning rate. A bad choice of the learning rate could mean a never-ending iteration and not reaching a minimum.

$$\Theta_0 = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \tag{2.7}$$

$$\theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) * x^{(i)} \tag{2.8}$$

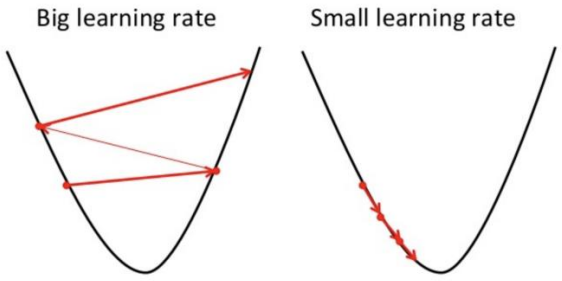

Fig. 2.3 Learning rate decision [9].

2.2.5.Overfitting

When evaluating the results of an applied algorithm, some mistakes can be found. Overfitting usually occurs when the hypothesis consists of a polynomial equation with too many degrees, and the curve goes through all the training examples present on a given dataset. Although it may seem as the perfect solution, as the cost function is close to zero, it is actually inaccurate.

This misreading of the hypothesis curve could be wrong if there are not many training examples or the curve is to complex. The presence of any new data will possibly make the algorithm fail to generalize. There are two common mistakes:

- Bias: The hypothesis curve is too simple and as a result, the data does not fit. There is a high error between the points and the training output.
- Variance or overfitting: The hypothesis curve fits almost perfectly the training examples and tend to fail to generalize.

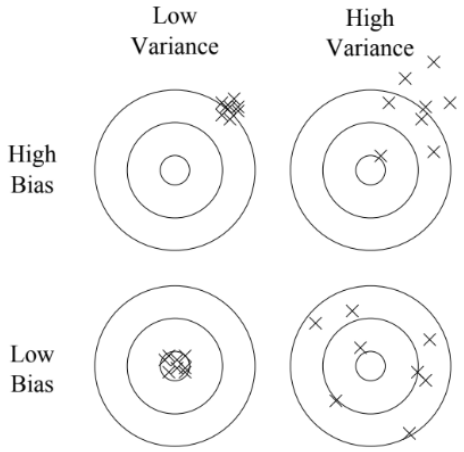

Fig. 2.4 Bias and variance [8].

Overfitting problem can be solved by reducing the number of features and introducing regularization.

2.2.6.Regularization

The first step towards solving overfitting is to regularize the cost function. This is achieved by adding a regularized term to the cost function, introducing a regularization parameter, λ.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right] \quad (2.9)$$

It is important to select a good value of λ, because if it is too large, the cost function will prone to underfitting. The objective of this new term is to adjust the weight that theta has on the cost function.

By convention [6], θ_0 is not penalized on the new term, that is the reason j starts at one and not zero.

This term is also included in the adjustment of θ on the gradient descent process, going from θ_1 to θ_n.

$$\theta_j = \theta_j - \frac{\alpha}{m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) * x^{(i)} + \lambda\theta_j\right] \quad (2.10)$$

Mathematically, the lambda, λ, effect on the optimization process could mean very little weight of some features. As it could imply, to reach a global minimum, small theta, θ, values that affects directly the X feature variable.

2.2.7.Feature Scaling

If the features present or extracted in the training examples are in different range of numbers, the contours of the model may become more elliptic, making it harder to select a valid learning rate, for instance, in the gradient descent process.

In order to solve this problem, feature scaling is needed. The general solution is to scale all the features between similar ranges. The acceptable ranges may depend on the problem that is being evaluated, by means of a mean normalization.

$$x_i = \frac{x_i - \mu_i}{s_i} \quad (2.11)$$

Having μ_i as the average value of x_i in the training set and s_i is the range of x_i , if a simple analysis is to be performed, or the standard deviation, in order to generalize the equation.

$$s_i = (max - \min)_{x_i} \text{ or } s_i = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x_i - \mu_i)^2} \quad (2.12)$$

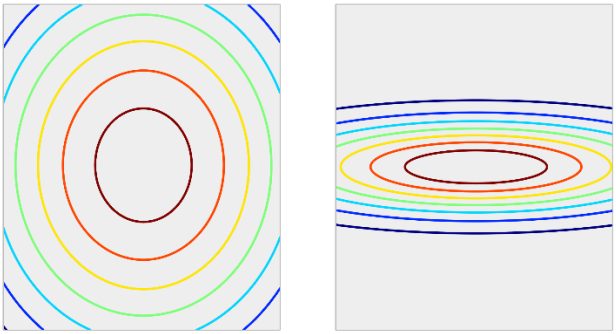

Fig. 2.5 Feature scaled, left vs before normalization, right [10].

2.2.8.Normal Equation

Rather than getting involved in an iterative process, there is the possibility of solving the value of θ analytically, using the so-called normal equation.

$$X = \begin{bmatrix} | & | & & | \\ x_0 & x_1 & \dots & x_n \\ | & | & & | \end{bmatrix}; y = \begin{bmatrix} | \\ y \\ | \end{bmatrix} \tag{2.13}$$

$$\theta = (X^T X)^{-1} X^T y \tag{2.14}$$

Having $(X^T X)^{-1} \in \mathbb{R}^{nxn}$. By applying this method, there is neither need to iterate nor to choose a learning rate α, which simplifies the process of optimizing theta. However, it is possible to encounter difficulties, such as slow processing times.

It is not a useful method when *n* is very large. Moreover, there are some matrices that do not have an inverse. If a non-invertibility situation is reached, it could be either due to the presence of redundant features, some features may be linearly dependent and it is advisable to review that it is not the case, or that there are too many features in comparison with the number of training examples $m \leq n$.

2.3. Machine Learning Algorithms

In this section of the state of art, the three algorithms that have been considered for the development of this thesis will be introduced. These algorithms are: logistic regression, neural network and Support Vector Machine. Later, they will be further explained in the survey section, and the decision of the chosen algorithm justified.

2.3.1.Logistic Regression

After a soft understanding of the linear regression algorithm, it will be evaluated the possibilities of logistic regression. This algorithm is mostly used for classification problems, composed of discrete valued outputs. In our further case, those will be zero and one, therefore, we can consider the problem as a binary classification problem:

$$y \in \{0,1\} \rightarrow \begin{Bmatrix} 0: Negative\ class \\ 1: Positive\ class \end{Bmatrix} \tag{2.15}$$

The hypothesis values are expected to be $0 \leq h_\theta \leq 1$, because this algorithm is based on the probability of a given example to be in a category. Through the computation of the sigmoid function and establishing a threshold value at which each class is selected, it can be determined the value of the output.

In the logistic regression model, the sigmoid equation is used as $g(\theta^T X)$ in the hypothesis:

$$h_\theta = \frac{1}{1+e^{-\theta^T X}} \tag{2.16}$$

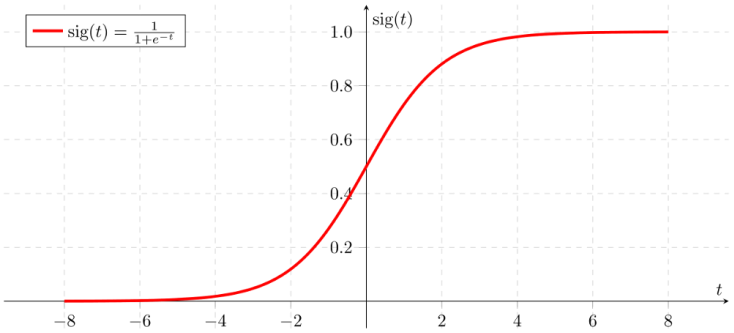

Fig. 2.6 Logistic regression h_θ representation [11].

It is also possible to develop a multiclass classification by means of this algorithm; it is called one-vs-all. To evaluate this problem, it is necessary to train a logistic regression classifier for each class, understanding the positive class as one and the rest of classes as zero.

2.3.2.Neural Network

Neural networks are a system that tries to represent the learning of the brain. It consists of an input and output layer and one or more hidden layers. The input layer is the feature vector and the output one is the hypothesis after the implementation of the algorithm is performed. For simplified calculation, every layer has a bias unit, which is a unit with value one. Hidden layers have weights in its units, being necessary that they learn to generalize the problem.

It can be used for binary classifications, where the output layer will consist of one unit, or multiclass classification problems, where the number of units in the last layer will be equal to the number of classes.

Having hidden layer adds computational cost, but it also allows the algorithm to process much more complicated features to extract a more precise insight for the hypothesis function.

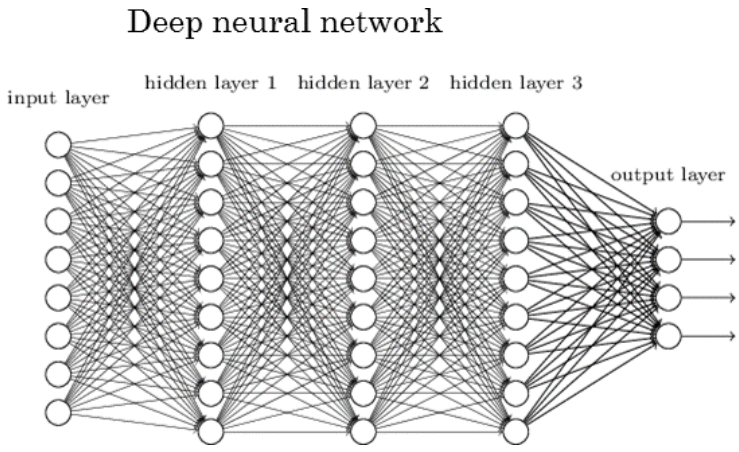

Fig. 2.7 Deep neural network [12].

2.3.3.Support Vector Machine

Support Vector Machine, SVM, is a large margin classifier. The objective of the algorithm is to develop a decision boundary that separates the different classes. It could work similar to a line with a margin separation between the classes or with more complex decision boundaries.

These decision hyperplanes could be of $\mathbb{R}^n$ dimensions. It introduces a new variable C for the development of the cost function in substitution of the lambda regularization parameter.

In order to adapt complex non-linear classifiers, it is necessary to introduce kernels. There are many types of kernels, from linear to gaussian, and it is not always obvious which type to introduce in the SVM system.

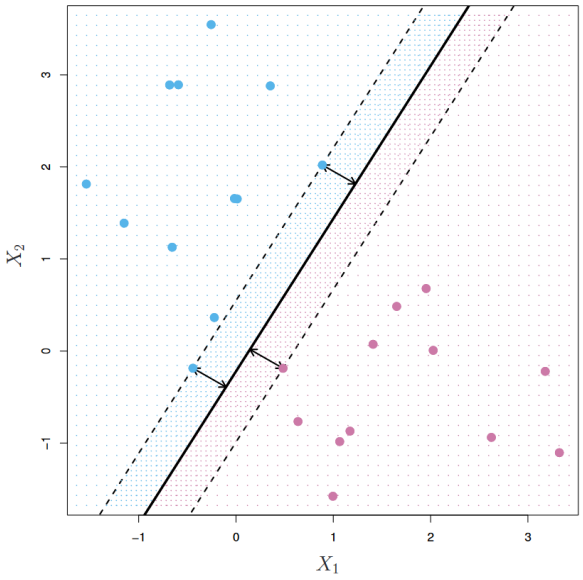

Fig. 2.8 Marginal SVM [13].

2.4. Natural Language Processing

As described by Gobinda G. “Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things” [14].

Therefore, it is understood that NLP is used to comprehend and use natural speech to perform tasks. Its applications vary enormously, though in the context of this bachelor thesis, it will only be evaluated the polarity analysis on sentences that can be performed by means of this method.

NLP has some issues along its analysis, regarding the meaning of the sentence to be examined or the knowledge of the words that compose said comment. More concisely, this bachelor thesis addresses one of the major issues to be known in speech analysis, sarcasm recognition.

Sentiment analysis is considered a subset of NLP. It is the technique used to understand the meaning of a given text, including the tone and polarity of what is being expressed by it. Nowadays, it is broadly used for speech recognition by companies and politicians, for instance. It allows these social actors to understand, from an extensive point of view, the general opinion that the common public has about their subject of interest.

This methodology is commonly developed employing deep learning techniques [15], as the evolvement of social media becomes a reality and the amount of data that can be collected into datasets increases and have an easier access.

Other algorithms, such as the logistic regression, neural networks and Support Vector Machines, can also be implemented to develop this analysis. It is necessary, in order to apply sentiment analysis, to preprocess the data into some new type that can be processed by the ML algorithm.

As Gobinda explains [14], some tools for developing such technique are related to the lexical and morphological analysis of the text, the semantic, word meaning and knowledge-based approaches and representation.

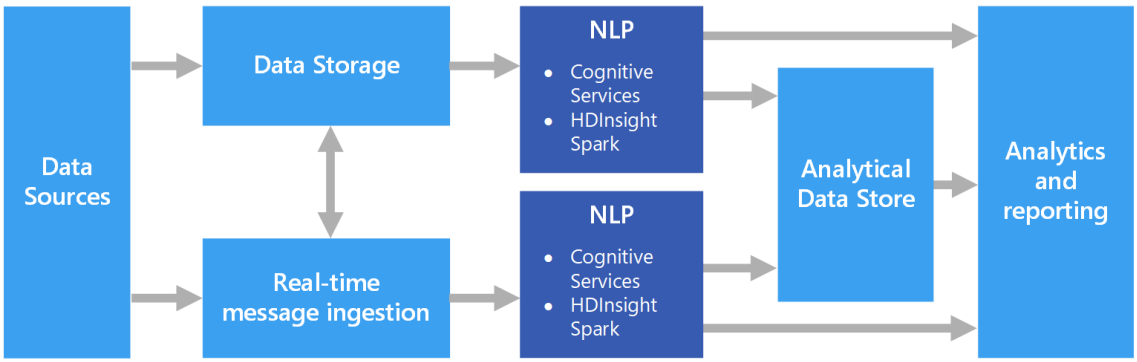

Fig. 2.9 NLP pipeline example [16].

2.5. Machine Learning Application in Sarcasm Recognition

In this section, other works regarding the topic covered along this final degree project will be analyzed and briefly explained. It is of great relevance the paper published by Mikhail Khodak, Nikunj Saunshi and Kiran Vodrahalli [17], as the dataset later used was created by them.

The first approach analyzed throughout the development of this bachelor thesis was published in 2011 by Roberto G. [18]. It emphasizes that the research done in the subject was beginning, due to lack of precise labeled comments that can feed the ML systems. The perspective of the work was based on tweets, and the dataset contained the URLs and *hashtags* employed in the comment.

The 2,700 tweets balanced corpus was built around three main classes, which were tagged to each comment by the researchers using the *hashtags* in the sentences: Positive (P), Negative (N) and Sarcastic (S). Therefore, they addressed the problem not only by the appearance of sarcasm on a given comment, but also combined it with the polarity of it. More precisely in the following comparison classifications: S-P-N, S-NS, S-P, S-N.

The features used were from different types. Lexical, using unigrams and bigrams based on dictionaries previously developed and with their weights based on positiveness. Polarity of the emoticons used was taken into consideration. Finally, the feature ranking expressed before. The algorithms proponed were Sequential Minimal Optimization and Logistic Regression. The SMO outperformed the results for the LogR algorithm. Having the best accuracy for the S-P-N of 57.22% and for the S-NS of 65.44%. Both using unigrams methodology.

Paolo Rosso has developed some papers surrounding the topic of irony and sarcasm recognition in text. Being of special interest their Twitter analysis on [19]. They performed the analysis on a 13,000 balanced dataset with tweets along the *hashtag #sarcasm*. It does study sentiment analysis and figurative language in the comments. Introducing the concept of making the sarcasm detection before determining the polarity of the sentence.

For evaluating the results, it was implemented cosine similarity and Mean-Squared-Error. Achieving a 0.689 statistical result for cosSim and 2.640 for MSE, considering all features. Being in second place at the SemEval-2015 Task 11, which first was 0.710 in cosSim. In a later approach, using Bag of Words, their results slightly improved.

Later, in 2017 Anukarsh and others [20] performed a sentiment analysis on a dataset composed of 2,000 balanced tweets performing a slightly different procedure. They divided their methodology in three phases. Starting with a pre-processing of the data, replacing hashtags, using a slang dictionary mapping, and using an emoji dictionary. Their following steps were to prepare the data, tokenize it, POS tagging, identifying the features and represent it. Finally, they performed the proper sarcasm detection algorithm.

The algorithms proposed for this paper were: Decision Tree, Random Forest, Gradient Boosting, Adaptative Boosting, Logistic Regression and Gaussian Naïve Bayes. And the results were evaluated only on the true positives.

There was performed three different distribution of training set and test set: 80:20, 70:30 and 60:40. Achieving their best effectiveness in the 60:40 split with an 81.82% of accuracy using Gradient Boosting. It is notable the efficiency obtained in the 60:40 distribution for the Logistic Regression classifier, with an accuracy of 40.25%.

That same year 2017, Dr. Kalpesh H. [21] proposed an algorithm based on techniques such as NB, Maximum Entropy, and SVM. Using a lexicon-based approach, as others did before, using already determined dictionaries to set the sentiment that is being transmitted with a short-text, but introducing a newly developed dictionary after the pre-processing of the data.

The architecture followed along this paper is introduced bellow.

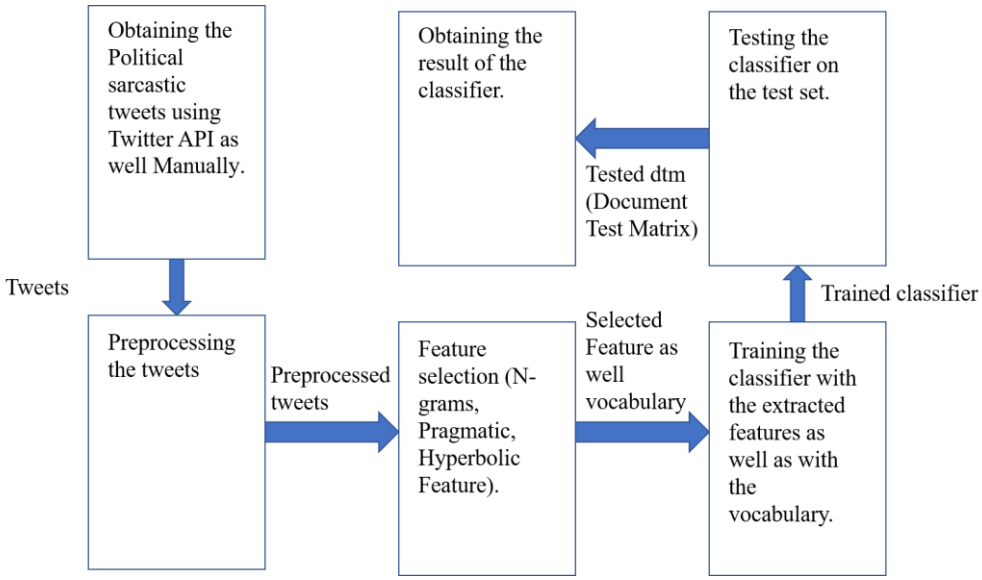

Fig. 2.10 Proposed architecture by [21].

The results of said analysis, were better that the current, at the time being, LogR and SVM. Achieving accuracies and precisions of 0.6875 and 0.8667 for SVM and 0.6667 and 0.9286 for LogR respectively.

Finally, *A Large Self-Annotated Corpus for Sarcasm* [17] is to be analyzed. It uses a balanced corpus of 1.3 million comments from a social platform called Reddit. The sarcasm label is determined by */s* that can be found in Reddit comments indicating one of the subreddits that it belongs to.

Being the largest dataset for the sarcasm recognition problem, it uses different approaches, such as, BOW, BOB, Sentence Embedding and comparing these analyses with the performance of human, both in average and majority. Using LogR and having the higher F1-score for bigrams of 75.8% and an F1-score for BOW of 73.2%. It does

consider the features appearing in the dataset, such as, sarcasm labeling, comment, subreddit, parent comment and more.

2.6. N-gram Analysis for Speech Recognition

N-gram analysis is considered to be the most manageable model of those that computes probabilities to concatenated words [22]. As a Language Model, it is a stochastic process model for word sentences.

It is relevant to speech recognition as it allows to create dictionaries of words. The probability calculations that follows the N-gram model can be used for predictive systems, helping the machine understand well-posed sentences. In the scope of this sarcasm recognition, N-gram can help understand the words or concatenated word that may imply the existence of sarcasm in a sentence, having a weight associated to each selected N-gram.

The probability calculations of the N-gram analysis, focusing on unigrams, bigrams and trigrams, having w_i as a concrete word, are:

- Unigram: $p(w_i)$ (2.17)
- Bigram: $p(w_i|w_{i-1})$ (2.18)
- Trigram: $P(w_i|w_{i-2}, w_{i-1})$ (2.19)

Generalizing to the N-gram probability for a word:

- N-gram: $P(w_i|w_{i-n+1}, \ldots, w_{i-1})$ (2.20)

3. ANALYSIS OF THE SYSTEM

In this chapter of the final degree project, the functional and non-functional requirements will be evaluated and described. Moreover, the algorithm evaluation by means of a survey will be developed.

3.1. Requirements

In this section, the requirements of the system will be defined and exposed. Starting by an introductory explanation of the tables that are to be used, defining each field that should be correctly filled.

- **Functional requirements**: Are considered to be those that specify the technical details of the system functionality.
- **Non-functional requirements**: These are considered to be the characteristics of the system.

The general table has the following form:

- **Identifier**: It is the identification code of the requirement of the form XR-YY.
  - R: corresponds to requirement.
  - X: will vary between functional and non-functional.
  - YY: will be the concrete number of the requirement of each list.
- **Name**: This field of the table gives a name to said requirement.
- **Priority**: Stablish the priority that has been given to a specific requirement in a three-step scale: low, medium and high.
- **Version**: Specifies the version number of the requirement.
- **Dependencies**: Indicates from which previous requirements this requirement is dependent.
- **Description**: Brief description of the requirement, indicating the finality that it has.
- **Need**: Three categorized level of need: optional, desirable and essential.

TABLE 3.1 FR AND NFR GENERAL TABLE

Identifier: XR-YY			
Name			
Priority	Low	Medium	High
Version		**Dependencies**	
Description			
Need	Optional	Desirable	Essential

3.1.1. Functional Requirements

Functional requirements of the system, along with their intercorrelations represented in the tables below.

TABLE 3.2 FUNCTIONAL REQUIREMENT FR-01

Identifier: FR-01			
Name	Extraction of archives .csv from an archive .csv		
Priority	Low	Medium	High
Version	1.0	**Dependencies**	-
Description	Extraction from the user comments from an archive of .csv extension to process the data		
Need	Optional	Desirable	Essential

TABLE 3.3 FUNCTIONAL REQUIREMENT FR-02

Identifier: FR-02			
Name	Extraction of .txt from .csv		
Priority	Low	Medium	High
Version	1.0	**Dependencies**	FR-01
Description	Creation of a .txt for an easier vocabulary library.		
Need	Optional	Desirable	Essential

TABLE 3.4 FUNCTIONAL REQUIREMENT FR-03

Identifier: FR-03			
Name	creation of a .txt library from the first porter-stemmer		
Priority	Low	Medium	High
Version	1.0	**Dependencies**	FR-01, FR-02
Description	Creation of the final vocabulary list.		
Need	Optional	Desirable	Essential

TABLE 3.5 FUNCTIONAL REQUIREMENT FR-04

Identifier: FR-04			
Name	Creation of .csv from .mat		
Priority	Low	Medium	High
Version	1.0	**Dependencies**	-
Description	Creation of a collection of sarcastic comments in .csv extension		
Need	Optional	Desirable	Essential

TABLE 3.6 FUNCTIONAL REQUIREMENT FR-05

Identifier: FR-05			
Name	Feature matrix as a .mat file		
Priority	Low	Medium	High
Version		**Dependencies**	FR-01, FR-03
Description	Development of the feature matrix that can be used in other implementations.		
Need	Optional	Desirable	Essential

TABLE 3.7 FUNCTIONAL REQUIREMENT FR-06

Identifier: FR-06			
Name	Model creation as a .mat file		
Priority	Low	Medium	High
Version		**Dependencies**	FR-03, FR-05
Description	Development of a variable that can be used in other implementations.		
Need	Optional	Desirable	Essential

3.1.2.Non-Functional Requirements

Non-functional requirements of the system, along with their intercorrelations represented in the tables below.

TABLE 3.8 NON-FUNCTIONAL REQUIREMENT NFR-01

Identifier: NFR-01			
Name	Programming language for the development of the interface		
Priority	Low	Medium	High
Version	1.0	**Dependencies**	-
Description	Algorithm is developed in Octave in order to have a free platform, strong in representation and math operations.		
Need	Optional	Desirable	Essential

TABLE 3.9 NON-FUNCTIONAL REQUIREMENT NFR-02

Identifier: NFR-02			
Name	Programming platform		
Priority	Low	Medium	High
Version	1.0	**Dependencies**	-
Description	MATLAB was used to ensure a correct lecture of the .csv archives.		
Need	Optional	Desirable	Essential

TABLE 3.10 NON-FUNCTIONAL REQUIREMENT NFR-03

Identifier: NFR-03			
Name	Programming language for the extraction of first .csv		
Priority	Low	Medium	High
Version	1.0	**Dependencies**	NFR-02
Description	MATLAB usage for the lecture and extraction of .csv files		
Need	Optional	Desirable	Essential

3.2. Evaluation of Learning Algorithm - Survey

In this section of the final degree project, the algorithms that are taken into consideration will be analyzed from a mathematical perspective, with the intention of developing a survey for the application of Logistic regression, SVM, Neural Networks in sarcasm recognition.

The main aspects to be studied along the explications of the learning algorithms will be their application range, hypothesis representation and cost function. Moreover, some distinctive feature that they may have will also be presented.

3.2.1. Logistic Regression

Logistic Regression algorithm is implemented for classification problems. It works from two classes, which is called binary classification problem, up to any number of classes that our algorithm may have to process, using a one-vs-all multiclass classification.

$$y \in \{1, \dots, k\} \rightarrow \begin{Bmatrix} 1\text{: } class\ 1 \\ 2\text{: } class\ 2 \\ \vdots \\ k\text{: } class\ k \end{Bmatrix} \tag{3.1}$$

Hypothesis

For the development of a hypothesis, logistic regression is a function that depends on the sigmoid function:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \tag{3.2}$$

This function allows to enclose the value of the hypothesis between zero and one. As can be seen in Fig. 2.6 Logistic regression h_θ representation . The hypothesis function in LogR calculates the estimated probability that the output of an example corresponds to a given class, which usually is determined by $y = 1$. Therefore, the probability of $y = 1$, given x and parameterized by θ is:

$$h_\theta(x) = P(y = 1|x\ ;\ \theta) \tag{3.3}$$

Considering that the class zero corresponds to the negative class, or the classes that are not taken into consideration in a multiclass classification, and one is the positive class, these is the basic property that this probability calculation should secure:

$$P(y = 0|x\ ;\ \theta) + P(y = 1|x\ ;\ \theta) = 1 \tag{3.4}$$

$$P(y = 0|x\ ;\ \theta) = 1 - P(y = 1|x\ ;\ \theta) \tag{3.5}$$

Decision Boundary

In order to determine which examples are positive and negative, a decision boundary could be implemented. The decision boundary is supposed to separate the classes setting conditions for each class. For the following example, being the features x_1, x_2 the axis, we can obtain the decision boundary:

$$x_1 + x_2 = 6 \tag{3.6}$$

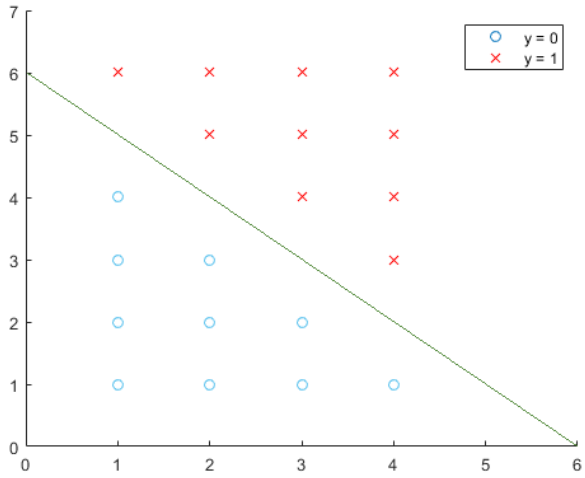

Fig. 3.1 LogR decision boundary example

A threshold value is determined, ε. Therefore, the conditions can be set, for instance, to predict $y = 1$ supposing a threshold value $\varepsilon = 0.5 \rightarrow h_\theta \geq 0.5$:

$$\theta^T X \geq 0 \tag{3.7}$$

And to predict $y = 1$ if $h_\theta < 0.5$:

$$\theta^T X < 0 \tag{3.8}$$

Cost Function

Regarding the cost function, there are major changes with respect to linear regression, with the substitution of terms and taking into account the MLE. The cost function should fulfill the condition:

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ y = 1 \\ -\log(1 - h_\theta(x)) & if\ y = 0 \end{cases} \tag{3.9}$$

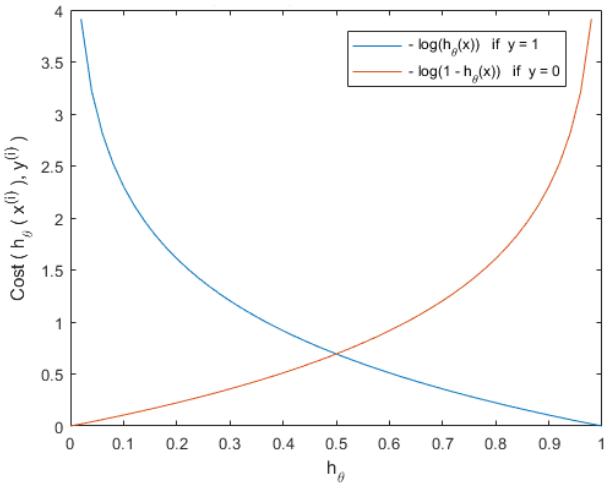

Fig. 3.2 Cost function representation [6].

In the figure we can observe that for $y = 1$ as the hypothesis gets closer to zero, the cost function tends to infinity and the other way around for the $y = 0$ case, giving the intuition that the error gets closer to zero as the hypothesis match with real output of the examples. Having the intersection between the two curves the threshold selected for the algorithm implementation.

The reasoning behind using this implementation instead of the least squared error implementation, used in linear regression, is that the function to be implemented can be convex or non-convex. If it is a non-convex equation, the algorithm implementation could reach a local minimum instead of the local. By applying this intuition, the global minimum is reached.

Having a training set $\{(x^{(1)}, y^{(2)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ the cost function of the LogR, using gradient descent, will be:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] \quad (3.10)$$

When determining the functions of theta weights that will be iterated, the partial derivative of the cost with respect of theta $\frac{\delta}{\delta\theta}J(\theta)$, will be:

$$\frac{\delta}{\delta\theta}J(\theta) = \sum_{i=1}^{m}\left(h_\theta(x)^{(i)} - y^{(i)}\right)x_j^{(i)} \quad (3.11)$$

Resulting in a continuous implementation of $j = 0$, where $x_0 = 1$:

$$\theta_0 = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta(x)^{(i)} - y^{(i)}\right)x_0^{(i)} \quad (3.12)$$

Finally, solving the potential problem of overfitting and generalizing the previous implementation for the *n* theta weights, regularization should be included. The cost function and theta refreshed values will be:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{m}\sum_{j=1}^{n} \theta_j^2 \quad (3.13)$$

$$\theta_j = \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x)^{(i)} - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}\theta_j^2\right] \quad (3.14)$$

There are other more complex optimization algorithms than gradient descent that could be applied, such as conjugate gradient, BFGS, L-BFGS, that for the sake of synthetization will not be included. Although it should be noted that they are faster than gradient descent for larger datasets and there is no need of choosing a learning rate.

A good optimization code for gradient descent using Octave/MATLAB for this and other learning algorithms would be using the *fminbnd* minimizer function.

3.2.2. Neural Network

Neural network is a very complex learning algorithm based on the functioning of the brain. Humans are capable of processing images because of the number of interconnected neurons that our brain have, from numbers to landscapes. Neural networks are capable of managing complicated operations through their hidden layers.

Introduction

Neural networks are composed of an input and output layer, along with a variable number of hidden layers. It typically consists of non-linear hypothesis. Some notation [6] that is going to be used along the explication of the functioning of this method is the following:

$a_i^{(j)}$: Activation of unit *i* in layer *j*.

$\Theta^{(j)}$: Weights matrix from layer *j* to layer *j+1*.

In the following figure, we can check that there are three layers. An input *x* layer with three input units, one hidden layer with three units and its activations *a*, and an output layer consisting of a hypothesis. Theta will be: $\Theta^{(j)} \in \mathbb{R}^{3x4}$.

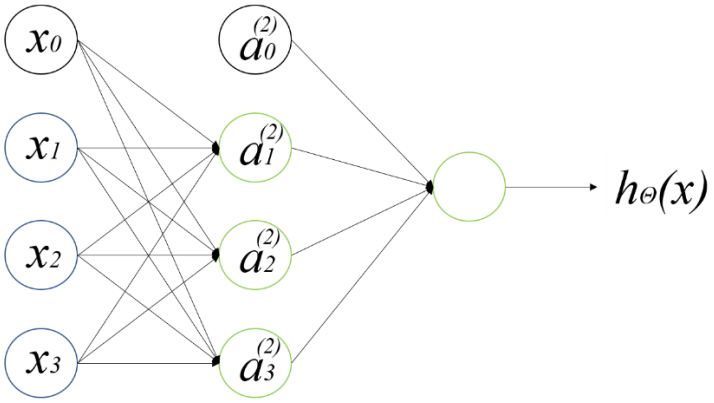

Fig. 3.3 Basic neural network representation.

By convention, a bias unit is added in each layer, they are not affected by the previous layer but affect the following. In the example, the hidden layer and output layer, will be computed:

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) \tag{3.15}$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) \tag{3.16}$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) \tag{3.17}$$

$$h_\Theta(x) = a_1^{(3)} = g\left(\Theta_{10}^{(2)}a_0 + \Theta_{11}^{(2)}a_1 + \Theta_{12}^{(2)}a_2 + \Theta_{13}^{(2)}a_3\right) \tag{3.18}$$

The dimensions of the $\Theta^{(j)}$ matrix will be determined by the number of units on each layer. If a network has s_j units in layer j and s_{j+1} units in layer j, the dimensions of the matrix will be $s_{j+1}\ x\ (s_j + 1)$.

Forward propagation is a procedure that optimizes the values of the features and weights going from the input layer to the output layer. Generalizing the formulas that will be used for this methodology are:

$$x = a^{(1)} \tag{3.19}$$

$$z^{(j)} = \Theta^{(j-1)}a^{(j-1)} \tag{3.20}$$

$$a^{(L)} = g(z^{(L)}) \tag{3.21}$$

$$g(z) = \frac{1}{1+e^{-\Theta X}} \tag{3.22}$$

Where, L is the total number of layers, s_L is the number of units of the output layer and if it is a multiclass classification problem $K > 2$, $s_L = K$.

Cost function

Taking into account that there are multiple layers, when analyzing the cost function, there would be more summations. The cost function will look alike the logistic regression one, considering the multiclass classification and the nested summations for the multiple nodes.

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - \left(h_\Theta\left(x^{(i)}\right)\right)_k\right] + \right.$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\Theta_{ji}^{(l)}\right)^2 \qquad (3.23)$$

The peculiarities of the cost function, the extra summations, are related to adding up the cost calculated for each value of the matrix in the program. As a side note, the *i* of the triple summation does not refer to the training examples.

Backpropagation algorithm

The backpropagation algorithm is a complementary way to forward propagation to compute the optimized thetas from the output node to the inputs, with different perspective to calculate the cost function and the gradient of that cost.

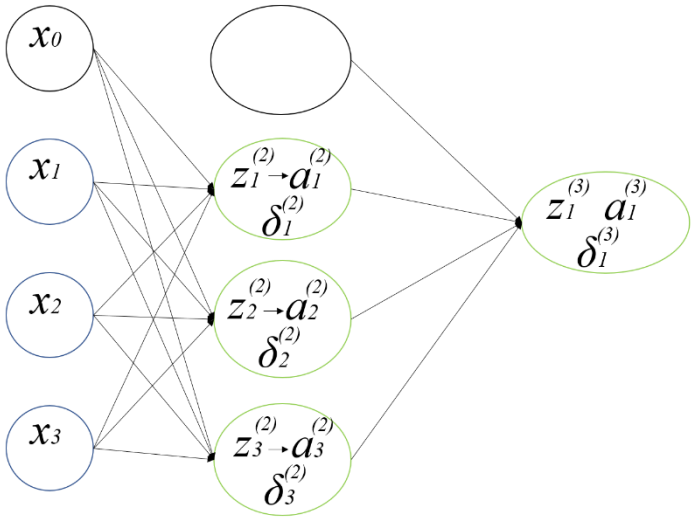

Fig. 3.4 Backpropagation algorithm and notation.

$\delta_j^{(l)}$: It can be considered the error of the node *j* in the layer *l*.

Before evaluating backpropagation algorithm, it is necessary to star by applying the forward propagation one and obtain the hypothesis.

Starting in the last node, where $a_i^{(l)} = \left(h_\Theta(x)\right)_j$, as the error at those units is possible to be calculated, because we know the value of *y*. Therefore, we can star by calculating $\delta_j^{(L)}$ as the subtraction of the activation of the current node of the last layer and the output real value *y*.

$$\delta^{(L)} = a^{(L)} - y \tag{3.24}$$

The calculation of the rest of the δ, knowing that "·" is an element wise multiplication, will be as follows:

$$\begin{gathered} \delta^{(L-1)} = \left(\Theta^{(L-1)}\right)^T \delta^{(L)} \cdot g'\left(x^{(L-1)}\right) \\ \vdots \\ \delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} \cdot g'\left(x^{(2)}\right) \end{gathered} \tag{3.25}$$

And there is no $\delta^{(1)}$ because the input layer is does not have error associated to it. And we obtain, without considering regularization:

$$\begin{gathered} \frac{\partial}{\partial \Theta_{ij}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)} \\ \delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \left(y^{(i)} \log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(h\left(x^{(i)}\right)\right)\right) \end{gathered} \tag{3.26}$$

3.2.3. Support Vector Machine

Support Vector Machine is a large margin classifies, meaning that there is a distance between the decision boundary found and the data that is given. Compared to LogR, SVM performs a slightly different cost function, as it can be observed in the following graph:

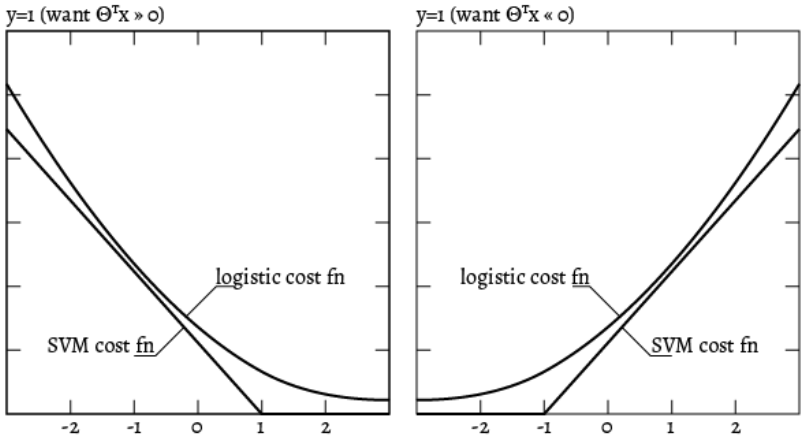

Fig. 3.5 SVM vs. LogR cost function representation [23].

Cost function and hypothesis

The function to optimize in an SVM does not take into account the number of training example, by convention explained in [6]. And the regularization parameter λ disappears, having a new term, C, in the first term. To have a better intuition of the functionality of this new term, it can be seen as $C = \frac{1}{\lambda}$.

$$\min_\theta C \sum_{i=1}^{m} \left[y^{(i)} cost_1\left(\theta^T x^{(i)}\right) + \left(1 - y^{(i)}\right) cost_0\left(\theta^T x^{(i)}\right)\right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 \tag{3.27}$$

The value of *C* parameter has a correlation with bias and variance. Large *C* values correspond to lower bias and high variance, whereas small *C* values work the other way around.

The hypothesis and decision boundary will be determined by:

$$h_\theta(x) = \begin{cases} 1 & if\ \theta^T x \geq 1 \\ 0 & otherwise\ \theta^T x \leq -1 \end{cases} \tag{3.28}$$

Having a range, between -1 and 1, with no concrete value. This is because the intention of SVM decision boundary is to be the most precise possible giving a maximum upwards and downwards margin. In the following figure, which is a comparison between SVM and LogR, this can be clearly distinguished.

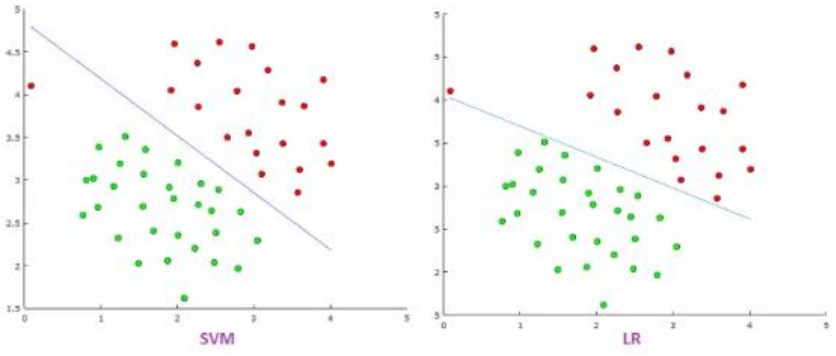

Fig. 3.6 SVM vs. LogR decision boundary [24].

From this figure it can be understood that LogR is more sensitive to the presence of values outside the common pattern, although, it should be noted that if necessary, with a large *C* value, SVM will capture too the outlier.

Math behind SVMs

Large margin classifiers are mostly based in the concept of the vector inner product. Some concepts must be introduced in order to proceed with the reasoning. It will be considered *p* as the projection perpendicular to the θ vector in the plane. Theta must be perpendicular to the hyperplane of the large margin classifier.

Therefore, the minimization of θ should be done:

$$\theta^T x^{(i)} = p^{(i)} \|\theta\| \tag{3.29}$$

$$s.t \quad \begin{matrix} p^{(i)} \|\theta\| \geq 1 & if\ y^{(i)} = 1 \\ p^{(i)} \|\theta\| \leq -1 & if\ y^{(i)} = 0 \end{matrix} \tag{3.30}$$

Kernels

In order to develop more complicated decision boundaries, it is necessary to simplify the SVM analysis. This can be done by the introduction of kernels. It is a way to overcome the computational expensive calculations of polynomials. The absence of kernel in an SVM algorithm is called linear kernel.

The main idea underlying kernels is to compute the proximity, f_a, of the points to some landmarks, $l^{(a)}$, previously determined. In the case of gaussian kernels, of the form:

$$f_a = \exp\left(-\frac{\|x-l^{(a)}\|^2}{2\sigma^2}\right) \tag{3.31}$$

So that,

- If $x \approx l^{(a)}$:

$$f_a = \exp\left(-\frac{\sim 0^2}{2\sigma^2}\right) \approx 1 \tag{3.32}$$

- If x is far from $l^{(a)}$:

$$f_a = \exp\left(-\frac{sth.\ large^2}{2\sigma^2}\right) \approx 0 \tag{3.33}$$

By introducing these kernels into the minimization function, it will be as follows:

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}cost_1\left(\theta^T f^{(i)}\right) + \left(1-y^{(i)}\right)cost_0\left(\theta^T f^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 \tag{3.34}$$

3.2.4. Previous Results Analysis on Sarcasm Recognition

Taking into account the research that has been developed, neural networks will not be considered for the forward development of the application algorithm for sarcasm detection in the thesis. Although they seem promising for the evolvement of irony detection systems, and having the best results in the field, the implementation algorithm commonly used are deep neural networks and deep convolutional neural networks, that exceed the scope of this bachelor thesis.

Given the following results table:

TABLE 3.11 ACCURACIES SURVEY

		SMV	LogR	SVM
S-NS [18]	Unigrams	-	60,72%	-
	LIWC*_F	-	59,83%	-
	LIWC*_P	-	63,17%	-
With Emoji and Slang dictionary [20]	80:20	-	40,29%	-
	70:30	-	38,33%	-
	60:40	-	40,25%	-
Without Emoji and Slang Dictionary [20]	80:20	-	41,03%	-
	70:30	-	39,89%	-
	60:40	-	32,03%	-
Lexicon-based approach [21]	Accuracy	68,75%	66,67%	-
	Precision	86,67%	92,86%	-
Bigrams [17]	F1-Score	-	75,8%	-
Balanced Dataset [25]	Accuracy	-	-	90,74%
	F1-Score	-	-	90,74%

It can be analyzed that logistic regression is a common election of algorithm for sarcasm recognition. However, its results are not generally promising. Only in the lexicon-based approach it obtained a high precision, with a good accuracy.

Therefore, and based on the impossibility of performing a deep neural network in the thesis; having set the scope of the final degree project in creating a simple solution; considering the mean of the accuracies of LogR; and in order to further develop the research of SVMs in the sarcasm recognition field, SVM will be the selected algorithm for this bachelor thesis.

4. DESIGN AND SYSTEM IMPLEMENTATION

In this chapter of the Bachelor Thesis, the design of the system will be presented, and the implementation explained and analyzed following a linear structure, from the beginning of the process to the results analysis.

4.1. Architecture of the System

The architecture of the system was made taking into account the decision of using an SVM implementation, the need of preprocessing the data of the corpus and the kernel decision. The architecture is the following:

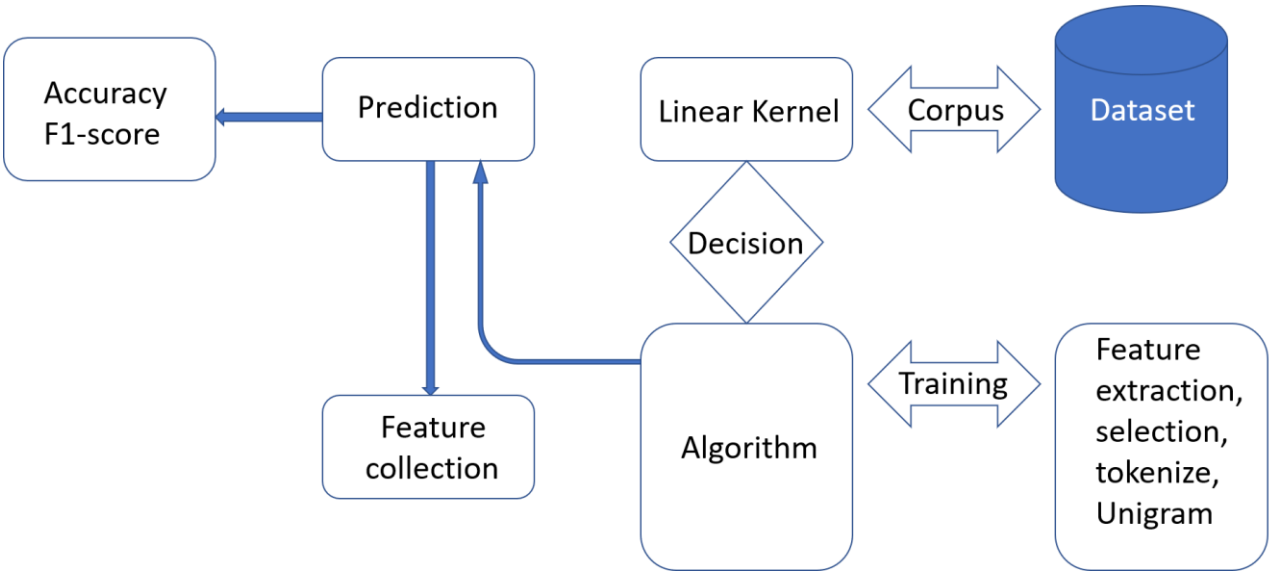

Fig. 4.1 Architecture of the SVM system.

4.2. Data

For this final degree project, the data was obtained from Kaggle datasets, although the source of the data is the work previously analyzed of [17]. It was chosen because the development of this work was very close to date, being of 2018, and initially thinking of using the complete dataset as a combination of training and test set for the algorithm implementation.

However, due to the computers available for the development of this thesis, it was not possible. When trying to compute the feature matrix, memory problems arise if the dataset being used was more than 40,000 sarcastic comments.

The data was generated by scraping Reddit platform, looking for the "\s" tag, that indicates sarcasm. It is composed of 1.01 million lines of *.csv* extension, divided into ten categories:

- **Label**: It has the values one or zero, being a binary classification problem, indicating if the comment is sarcastic or not respectively.
- **Comment**: The comment that is to be analyzed.

- **Author**: The username of the creator of the comment.
- **Subreddit**: This social media platform is organized around subreddits, that indicates the topic that is being treated under a thread.
- **Score**: Value set to a comment.
- **Ups**: The amount of upvotes that a post have received from other users.
- **Downs**: The number of downvotes that a post have received from other users.
- **Date**: Publication date of the comment on the platform. It only includes the year and month of creation.
- **Created**: Not only the date of creation but also the hour, minute and second.
- **Parent comment**: The comment that precedes the one that is being evaluated for the presence of sarcasm.

TABLE 4.1 EXAMPLE OF ONE TRAINING EXAMPLE

Label	*0*
Comment	*You don't have to, you have a good build, buy games or save it*
Author	*SoupToPots*
Subreddit	*pcmasterrace*
Score	*1*
Ups	*-1*
Downs	*-1*
Date	*2016-10*
Created	*27/10/2016 19:11*
Parent comment	*"What to upgrade? Have $500 to spend (mainly because it's my birthday on the 31st) and I'm not really sure what to upgrade. I don't have to spend all $500, I could spend as little or all of it if I want. Specs are: CPU: i5-6600K Cooler: CM 212 EVO RAM: 8GB Corsair Vengence LPX 2400MHz GPU: EVGA GTX 1070 SC Case: Corsair Spec Alpha PSU: EVGA P2 650w Storage: One 480 GB Corsair Force LE Monitor: BenQ XL2411Z Keyboard: Razer Blackwidow Tournament Edition What I'm thinking of right now is: -Get a 1TB WD Blue -Buy another 8GB of RAM -Replace my Razer Blackwidow with a keyboard with actual Cherry MX switches -Replace my case (Doesn't really offer features for good cable management) -Save up for a Volta Card"*

The data is balanced, meaning that there is approximately the same amount of positive and negative examples, making it easier for the implementation of a good learning algorithm if the whole data is to be evaluated. It is structured approximately all of the zeros labeled in the first half of the dataset and all of the ones in the second half.

4.3. Simplifications

One of the main objectives of the bachelor thesis is to try to simplify the problem as much as possible, trying to achieve an acceptable accuracy. In order to make it more accessible to the common public and try to make it computationally less expensive to compute, only the first two columns of the dataset are to be considered for the implementation process, the label and the comment.

With this simplification some interesting insights will be lost, as the parent comment could be important for the prediction. Moreover, the subreddit that they belong to could be interesting to analyze, because having it as a feature could allow the classification problem and the weights associated to the library to be dependent to the topic that is being studied. Having a more topic-based solution for the speech recognition problem and being able to develop more accurate solutions for the irony detection.

Along with the creation of the dictionary, only the words that are repeated at least five times are taken into consideration for the feature vector. This simplification will reduce the accuracy of the problem but decrease by far the computational time that the code will take to execute.

Considering the computing cost of the code, the library will have another simplification, a Porter Stemmer will be implemented in order to group the words that have the same stem, this simplification will be further explained in the next section.

4.4. Data Preprocessing

In order to correctly classify the data of the sarcasm corpus made by comments, it is necessary to make some preprocessing on it. The computers available for the research project cannot compute that large amount of data regarding memory shortage. In order to solve that, the data must be structured alternating positive and negative values. Moreover, the data should be structured between the acceptable ranges that the computer can process.

- *Extract_sarcasm* function: This function orders through a loop the comments alternating between ones and zeros, the reasoning is that with this implementation, it is possible to extract any amount, easily, of comments balanced. Moreover, subtracts the datasets that are going to be implemented to compare the results.

  Finally, the new balanced short datasets will be written in *.csv* and *.txt* files. This way, the new shorten examples can be fed to the algorithm. The *.csv* file will be used to extract the features and make a unigram analysis, whereas the *.txt* file will be used for the development of the dictionaries.

- *CorpusProcess1* function: Is a function that develops a Porter stemmer on each word of the input comments, writing them into a .txt files in order to create a library. Each non-repeated word will be saved in the file in a different row.

  The reasoning behind this function is to lower all the words in the comment, eliminate the emojis and punctuation. By doing so, the accuracy of the algorithm may be reduced. Sarcasm in social media is exposed commonly with the alternation of lower case and capital letters, hyperbole use, punctuation signs and numbers. However, eliminating these characters, the algorithm becomes more accessible and computationally less expensive.

Although time consuming, as one of the objectives of this bachelor thesis is to develop the whole algorithm, the creation of a library is fundamental for the rest of the implementation.

14	number
15	for
16	they
17	but
18	this
19	have
20	are
21	not
22	just
23	on
24	with
25	he
26	be
27	was
28	so
29	can
30	because
31	at
32	all
33	no
34	like
35	if
36	we
37	what

Fig. 4.2 Vocabulary list

- *CorpusProcess2* function: Having a new dictionary of words, that does not include punctuation such as double quotes, which affected the overall accuracy of the system, as they were the most common word. Again, porter stemmer is applied and with the singularity that this time there would be a comparison between the words of the comment and the dictionary using the expression *strcmp*, string compare.

```
wow it is totally unreasonable to assume that the agency that covered up
 bush war crimes because muh republican party would be partisan as fuck
```

Fig. 4.3 Porter Stemmer example

A feature matrix is constructed, having as much rows as the training set comment that are fed to the program and the number of columns corresponding to the amount of words of the dictionary. As a comment is being analyzed, there is a comparison of each word, and if it appears in the dictionary, the position of that row comment in the column of that word will be one.

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	1	0	0	1	0	1	0	1	1	0
5	1	0	0	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0	1	1	0
7	1	0	0	0	0	0	0	0	0	0
8	0	1	1	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0
11	0	1	1	1	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	1	0
14	0	0	1	0	0	1	1	1	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	1	0	1	1	1	0	0	0	0
18	1	1	0	0	0	0	0	0	0	0
19	0	0	1	1	0	0	0	0	0	0
20	1	0	1	0	0	0	1	0	0	0
21	0	1	1	0	0	0	0	0	0	0

Fig. 4.4 Feature variable

This is achieved by the development of an index vector that gets the word position in the dictionary of each comment.

```
comment_index =

   193     5     9   158     3     7     1     7    63   313    30   231    43    26    41   131
```

Fig. 4.5 Comment indexing example

4.5. Algorithm Implementation

For the algorithm implementation a linear kernel was chosen after comparing that the results did not varied significantly compared with the use of a gaussian kernel. The structure of the code followed the next structure:

- *Train* function: In this function the training of the algorithm is performed, using an SVM using a *C* value of 0.01. In return, it is obtained an implemented model stored as a *struct* type variable in the workspace that contains the X and y variables, the kernel if needed, and more.

- *Predict* function: This is the function that will find the outputs expected for the training set and the test set. It receives the information of the model, and as an input of the function does also have the feature matrix X. The probability calculations happen in this function.

  The output of this function consists of a column vector that indicates the class that each comment corresponds to. With these calculations, and having the original labeled data, it can be evaluated the algorithm.

```
yeah            (1.096957)
because         (0.742140)
dropped         (0.700000)
obviously       (0.629879)
no              (0.623472)
could           (0.618414)
all             (0.601772)
guy             (0.595300)
right           (0.582374)
game            (0.574866)
women           (0.556058)
wow             (0.554384)
forgot          (0.500000)
white           (0.472703)
yea             (0.468038)
```

Fig. 4.6 Vocabulary most weighted words for sarcasm

4.6. Evaluation Development

The evaluation of the algorithm has been developed by different methods in the function *Results*. Firstly, calculating the general accuracy on the training set that the model is returning, along the prediction variable. Some concepts need to be introduced in order to present the results evaluation.

Accuracy

Through the development of this bachelor thesis, the accuracy is going to be considered as the percentage of accurate predictions over the total number of examples evaluated by the set:

$$Acc = \frac{predTrue}{number\ of\ examples} \tag{4.1}$$

$predTrue$: Prediction labels that are equal to y label.

y: Dataset label.

Precision and Recall

Precision and recall are two different methods for evaluating a ML algorithm. Ideally the objective is to have high values for both, although as precision increases recall decreases and the other way around. Precision is evaluated as the number of true positives over the number of predicted positives, whereas recall is evaluated as the number of true positives over the actual number of positives. The notation that will be followed is:

TABLE 4.2 TRUTH TABLE FOR EVALUATION OF THE ALGORITHM

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

TP: True Positive. FP: False Positive.

TN: True Negative. FN: False Negative.

The formulas that will be applied for both concepts are:

$$Precision = \frac{TP}{TP+FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.3}$$

$Prec$: Precision. Rec: Recall.

F1-Score

F1-Score is an additional method to obtain the accuracy of a ML system. It is considered to be a good indicative of the performance of said algorithm, as it is computed with the combination of precision and recall. It is a good method to analyze skewed data, that is not balanced between the different classes.

$$F1 = 2\frac{Pred \cdot Rec}{Pred+Rec} \tag{4.4}$$

5. RESULTS AND EVALUATION

In this section of the final degree project, the results and the variations in the algorithm and data configuration will be presented. Later, the performance of said algorithm will be evaluated and a decision over which configuration should be selected for future works made.

5.1. Test Structure

The data, aforementioned in section 334.2, is being separated into two sets, as it is commonly done in ML applications. A training set that usually represents 80% of the total data to be analyzed and a test set, to check if the model fits the rest of data, of a 20%.

In order to analyze the impact of the size of the dataset, different configurations have been tested. The size of each of the training sets have remained constant along the different configurations, but the test set size has been modified, as there is more data than the one used in these configurations. To sum up, there are six training set sizes and each training set is evaluated in three test set configurations: 80:20, 70:30, 60:40, as it can be seen in the following table, where the amount of data of each analysis is presented.

TABLE 5.1 SIZE OF THE DIFFERENT DATASET CONFIGURATIONS

#	Training set	Test set (80:20)	Test set (70:30)	Test set (60:40)
1	1600	400	686	800
2	4000	1000	1715	2000
3	8000	2000	3429	4000
4	12000	3000	5143	6000
5	16000	4000	6858	8000
6	24000	6000	10286	12000

By analyzing said configurations, it can be observed how the different kinds of accuracies evolve, showing the best performance that the system, with the features and parameters selected, can have.

The decision of which dataset configuration to select for future works, if the computers available have the same memory capacity, will be determined by a combination of the accuracy of the training set and test set, and the precision, recall and F1-score over the test set.

5.2. Results Presentation

In this section, the results obtained under the different dataset sizes will be presented, and an analysis performed over each of the data. In said analysis, it will be evaluated the distribution of the test set over the different configurations.

The table with the results obtained after performing each implementation is presented below:

TABLE 5.2 RESULTS

Training set	Training/Test	Training Accuracy	Test Accuracy	Precision	Recall	F1-Score
1600	80:20	71.25%	60.50%	62.35%	53.00%	57.30%
	70:30		58.75%	60.00%	52.48%	56.00%
	60:40		58.88%	60.11%	52.75%	56.19%
4000	80:20	70.18%	64.70%	68.33%	54.80%	60.82%
	70:30		63.62%	67.11%	53.33%	59.43%
	60:40		63.40%	66.71%	53.50%	59.38%
8000	80:20	71.13%	65.15%	69.75%	53.50%	60.55%
	70:30		64.57%	69.06%	52.74%	59.81%
	60:40		64.63%	69.03%	53.05%	60.00%
12000	80:20	71.73%	63.87%	67.25%	54.07%	59.94%
	70:30		63.58%	66.65%	54.34%	59.87%
	60:40		63.65%	66.89%	54.07%	59.80%
16000	80:20	71.58%	64.28%	68.65%	52.55%	59.53%
	70:30		64.38%	68.87%	52.46%	59.56%
	60:40		64.38%	68.77%	52.68%	59.65%
24000	80:20	71.38%	64.95%	69.31%	53.67%	60.49%
	70:30		65.19%	69.57%	53.98%	60.79%
	60:40		65.38%	69.86%	54.08%	60.97%

It can be observed that generally, over each of the different configurations and performance analysis, the best result is obtained in the 80:20 data configuration. The ranges of the results over each of the training set data is similar, therefore it is not necessary to analyze 80:20, 70:30, 60:40 configurations individually. As it can be seen in Fig. 5.5 Performance ratios cluster graph in test set.

5.3. Results Evaluation

5.3.1. Training Accuracy

Training set accuracy over the different scenarios is very similar, indicating that the general performance of the SVM is successful. Moreover, as an insight, it can be observed that the size of the training set does not influence the performance under said analysis. The mean training accuracy is of 71.21% and the maximum is achieved under the 12,000-training set size configuration, being 71.73%.

To improve this accuracy, other features must be taken into consideration. Probably, considering a slang and polarity dictionaries as well as an emoji dictionary could be useful.

Moreover, making the system more complex, without applying a Porter stemmer algorithm, could include more cases of sarcasm, as this will bring back punctuations, quotations and numbers.

5.3.2. Test Accuracy

The training set versus test set configuration of 80:20 is selected to analyze this performance ratio. The mean accuracy is 63.55% and the maximum is achieved under the 24,000-dataset configuration, being of 65.38%. The results considering the smaller dataset are poor, although as the number of training examples gets bigger, the accuracy of the test set increases.

This gives the insight that incrementing the dataset size, could be beneficial for the performance of the algorithm on not labeled data. The reasoning behind this insight derives from the amount of new words that would appear and a readjustment of the weights for all of the words in the dictionary.

TABLE 5.3 MEAN TEST ACCURACY

Training Data	Mean Test Accuracy
1600	59.38%
4000	63.91%
8000	64.78%
12000	63.70%
16000	64.35%
24000	65.17%

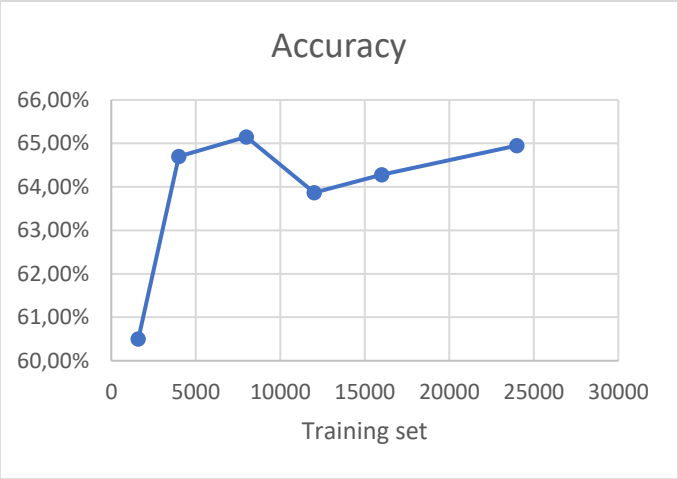

Fig. 5.1 Accuracy graph in test set.

5.3.3. Test Precision

The training set versus test set configuration of 80:20 is selected to analyze this performance ratio. The mean accuracy is 67.13% and the maximum is achieved under the 24,000-dataset configuration, being of 69.86%. However, the 8,000-dataset analyzed do have a similar precision.

Again, it seems that increasing the dataset size could be beneficial for making a more successful implementation. It is possible that the ups and downs are related to small groups of comments that may appear along as the number of examples considered increases and affect negatively when they are not in a large enough dataset to not be relevant.

TABLE 5.4 MEAN TEST PRECISION

Training Data	Mean Test Precision
1600	60.82%
4000	67.38%
8000	69.28%
12000	66.93%
16000	68.76%
24000	69.58%

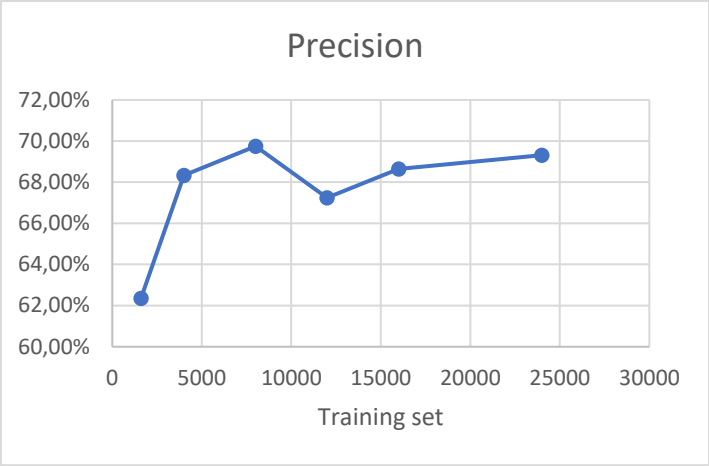

Fig. 5.2 Precision graph in test set.

5.3.4. Test Recall

The training set versus test set configuration of 80:20 is selected to analyze this performance ratio. The mean accuracy is 53.39% and the maximum is achieved under the 24,000-dataset configuration, being of 54.34%. The results in the recall section are common, although they reveal that there is a proportionally large number of false negatives, so that the threshold value selected for the algorithm could be better.

TABLE 5.5 MEAN TEST RECALL

Training Data	Mean Test Recall
1600	52.74%
4000	53.88%
8000	53.10%
12000	54.16%
16000	52.56%
24000	53.91%

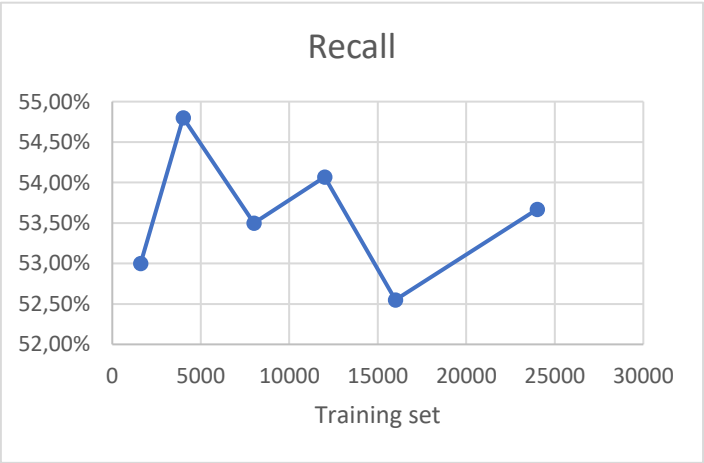

Fig. 5.3 Recall graph in test set.

5.3.5. Test F1-Score

The training set versus test set configuration of 80:20 is selected to analyze this performance ratio. The mean accuracy is 59.45% and the maximum is achieved under the 24,000-dataset configuration, being of 60.97%.

The decision of which algorithm to choose will be determined mostly by this performance ratio, as it is a combination of the others. It has small variance between the different dataset configurations, seemingly having reached the best overall performance that the algorithm, as it is configured can achieve.

TABLE 5.6 MEAN TEST F1-SCORE

Training Data	Mean Test F1-Score
1600	56.50%
4000	59.88%
8000	60.12%
12000	59.87%
16000	59.58%
24000	60.75%

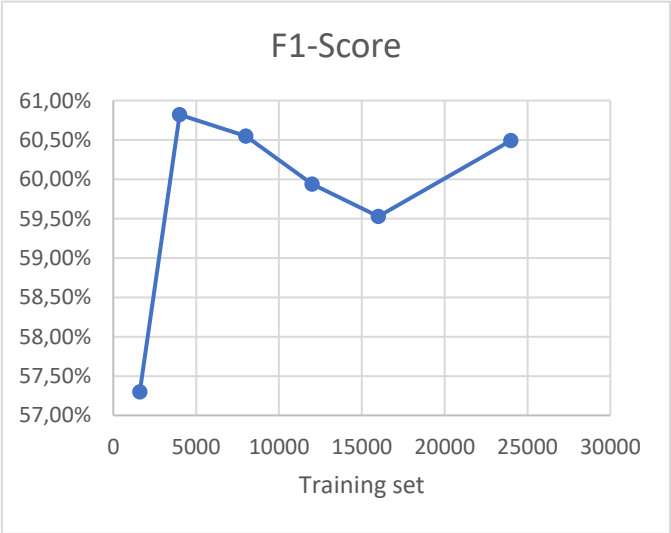

Fig. 5.4 F1-Score graph in test set.

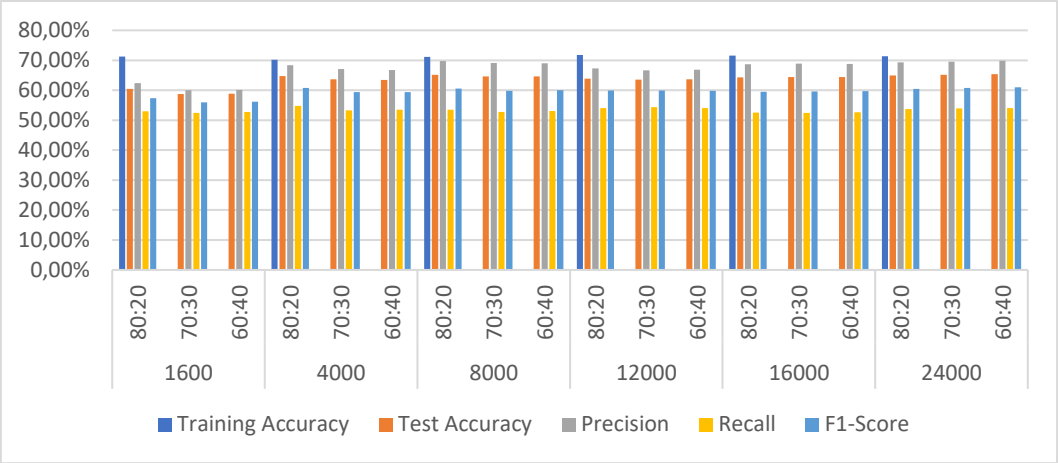

Fig. 5.5 Performance ratios cluster graph in test set.

In the overall cluster graph, it can be observed that the ratios between the different configurations maintain the same ranges and makes slight and persistent increases in performance as the dataset size increases.

5.3.6. True Positives, False Positives and False Negatives

The data related to the true positives, false positives and false negatives along the different scenarios is presented in the table below. As before said, the proportion between these three parameters is approximately constant.

TABLE 5.7 TP, FP, FN

Training Data	Training/Test	TP	FP	FN
1600	80:20	106	64	94
	70:30	180	120	163
	60:40	211	140	189
4000	80:20	274	127	226
	70:30	457	224	400
	60:40	535	267	465
8000	80:20	535	232	465
	70:30	904	405	810
	60:40	1061	476	939
12000	80:20	811	395	689
	70:30	1397	699	1174
	60:40	1622	803	1378
16000	80:20	1051	480	949
	70:30	1799	813	1630
	60:40	2107	957	1893
24000	80:20	1610	713	1390
	70:30	2776	1214	1390
	60:40	3245	1400	2775

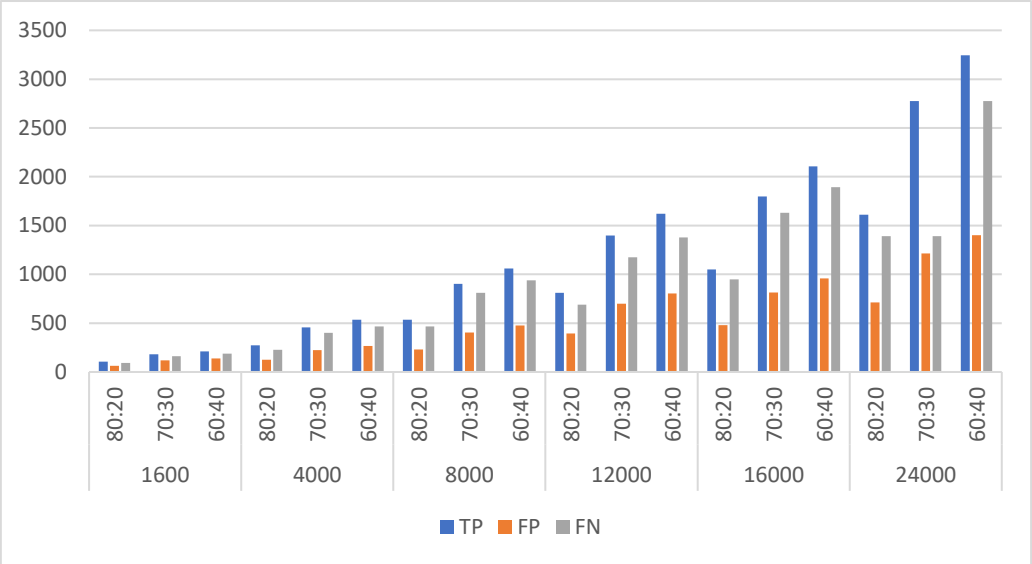

Fig. 5.6 TP, FP and FN cluster graph in test set.

The amount of false positive and negatives is significant, being remarkably bigger the number of false negatives, between 25-50%. In order to change that, the threshold for the large margin classifier should be rearranged. So that, more cases are considered as positive.

6. PROJECT MANAGEMENT

In this chapter, the planning that has been followed along the development of this final degree project is presented, including the estimated and real hours that has been dedicated to each specific chapter. With this planning it is expected to make understandable the process of creation that this work has followed.

Moreover, the budget estimation is introduced, analyzing human and resources associated costs. This will have a direct implication with the socioeconomic impact described in the introduction.

6.1. Planning

The planning of this work can be divided in three stages: Study phase, analysis and design phase and document creation phase.

- **Study phase:** The knowledge regarding this topic of study was limited at the beginning of the project. Therefore, understanding the theorical prospective an careers possibility regarding ML applications, it has been essential a deep study of the field.
    - **General ML study:** By means of online courses and technical documentation a deep understanding of ML basics was expected and achieved.
    - **Papers location, reading and study:** The survey analysis started with an understanding of the state of art implementations that had been carried out in recent years. By means of this study, it was expected to find improvements possibilities to be developed in this bachelor thesis.
    - **Algorithms study:** Advanced concepts understanding regarding the three algorithms considered along the survey process was to be processed.
    - **Octave/MATLAB learning:** Having a basic knowledge regarding the programming language that were to be used, it has been necessary a profound study of the platforms and pieces of code that could be implemented throughout the development of the code.
- **Analysis and design phase:** In this phase it was expected to develop the architecture of the system and code the entire algorithm.
    - **Architecture:** The architecture of the system was designed, understanding the functionality that the whole system should have and the main components to be developed.
    - **Preprocessing of the data:** Along this process, some transformations of the data were required. Regarding the way it introduced into the program and the separation of the different datasets.
    - **SVM implementation:** The code for the Support Vector Machine was created and tested in order to see the possible failure situations that it may encounter.

    - **Results development:** The analysis of the performance of the system was developed.
    - **Code improvement:** From an iterative perspective, as the code was being written, possible improvements were introduced and analyzed organically.
- **Document creation phase:** The final phase of the process consisted in reflecting the work done into a document.
    - **Results visualization:** The visualizations of the results are essential in order to have a better understanding of the performance of the system and to present the results in accessible illustrations and tables.
    - **Bachelor thesis writing:** Finally, the writing of the thesis was performed, analyzing the different chapters that are expected to be present.

In the following table, the hours dedicated to the bachelor thesis are presented, with a comparative between the expected hours and the real hours of each of the aforementioned phases.

TABLE 6.1 HOURS PLANNED

		Number of hours	
		Estimated	**Real**
Study Phase		300	395
A	General ML study	250	350
B	Papers location, reading and study	10	15
C	Algorithms study	30	20
D	Octave/MATLAB learning	10	10
Analysis and Design Phase		54	117
E	Develop architecture	3	3
F	Develop Preprocessing of data	6	25
G	Develop SVM implementation	20	26
H	Results development	20	55
I	Code improvement	5	8
Document Creation Phase		63	89
J	Results visualization	3	2
K	Bachelor thesis writing	60	87
Total		**417**	**601**

6.2. Budget

For the estimated budget, it has been considered the human and resources associated costs, including hardware and software costs.

For the human associated costs, it has been assumed that only two wages are to be considered. The student cost will be the one represented by the annual mean wage of a junior engineer, that according to Ingeniariak [26], is of 29,789€ before taxes. Using the same reference, the annual mean wage associated to the thesis supervisor of this project, corresponding to a senior engineer wage, is of 48,132€.

Therefore, the total cost of the project associated to human resources is presented in the following table:

TABLE 6.2 COST ASSOCIATED TO WAGES

	Hours	**Hourly wage**	**Cost Project [€]**
Junior Engineer	601	14,89	8951,59
Senior Engineer	20	24,07	481,32
Total cost [€]			**9432,91**

When analyzing the resource associated costs, hardware and software are evaluated. In hardware analysis, the computer used is the only resource to be considered, as the writing materials along with the rest of the cost can be ignored, as well as the energy cost.

Software analysis considers the cost of the platforms used. MATLAB license is a peculiar case, as it is expensive, but as university allows its usage cost free, it is considered to be zero. Only the office package used for the document writing and images creation is considered.

The total cost of the resources used is presented in the table below:

TABLE 6.3 RESOURCES COST

Resource	**Cost [€]**	**Lifespan [yrs.]**	**Use**	**Depreciation cost**
Computer	900	5	5 Months	75
MATLAB	0	-	4 Months	0
GNU Octave	0	-	5 Months	0
MS Office Package	99	1	3 Months	24,75
Total Resource Cost [€]				**99,75**

Adding up, the budget of the project should be: 9532,66€.

6.3. Socioeconomic Impact

As commented in the introduction 1.5, The socioeconomic impact of this technology is to be further explained. It will be explained marketing, energy and politics scopes.

- From marketing perspective, speech recognition is of great importance in order to focus the economic resources in the correct public and product. Companies can get the insight of the common public opinions regarding a product that has been launched to the market recently and what to improve for next generations products.

- Politicians use speech recognition in their campaigns to understand what methodologies that are being implemented are effective and the population sector where their aims are more probable to have beneficial effects for them.

- Energy sector in Spain is living a stage of increase in the general price of electricity. Moreover, nuclear power plants are under nuclear moratorium, meaning that the processes of construction and renewed power plants are stopped. This is due to the concerns regarding the safety of this power plants. For instance, if a study is to be performed in the general opinion regarding the electric bill, which has previously occurred by REE, speech recognition can help identify the main problems to be solved, making it probable that the efforts are allocated towards them, instead of focusing the attention in other aspects.

Summing up, in all this fields, speech recognition plays a relevant role to the new developments of strategies. Social media is one of the strongest platforms where people express their opinions as reflected previously in 1.1. Sarcasm recognition algorithms, supported by speech recognition, are capable of producing augmentations on the effectiveness of said algorithms, being a strong tool to consider in the following years to come, with great economical prospective.

However, even though the results of the algorithm have been successful, and it has achieved an interesting level of performance, this accuracy sarcasm detection system is not enough to be commercialized, as some deep neural networks have better results. For that purpose, the algorithm should be improved.

7. CONCLUSIONS

In this chapter, the technical conclusions of the project will be developed, as well as explained the competences that have been achieved and the future works that could follow this bachelor thesis.

7.1. Technical Conclusions

When analyzing the conclusions of the project, they will be evaluated taken into consideration the objectives that were proposed in the introduction chapter 1.3. Each point will be evaluated individually.

With respect to the survey, it can be concluded that the objectives were fully accomplished. A profound study along Machine learning competences and the different algorithms that are to be considered has been performed successfully. The selection of an SVM was based on the previous works developed and their performances.

Moreover, the implementation of the code has been made originally from the beginning, with little contributions of other authors. With the creation of a new dictionary that contained the word in the comments of the corpus.

It should be remarked that due to the limited resources for the execution of the code, it was not possible to scale the project to the whole dataset, which would have given a new perspective of the performance of the algorithm and developed a more sophisticated model.

Moreover, the time of processing of each distribution of the dataset, did not allow to try more configurations. Although the results of the configurations tried are successful, some variations of parameters could have been assessed.

Finally, another main objective of the bachelor thesis was to try to make more accessible the algorithm. This was achieved by means of the survey and simplifying some of the features to make the code more understandable. Therefore, it can be considered that this aim has been accomplished.

7.2. Competences

In this chapter, the competences used and achieved along the development of this final degree work will be exposed and explained. Regarding the competences used, they can be presented as:

- Programming capacities achieved in numerous degree subjects along these past years, regarding *C*, MATLAB. Used for the code writing.

- Project management capacities to organize the project in its different stages, knowing which parts will be more time-consuming.

- Supported on economical subject coursed these years, the socioeconomic impact that the project could have, has been evaluated, as well as the budget.

The competences that has been achieved throughout the development of this work, are presented below:

- ML general knowledge accompanied by mathematical models for the development of the algorithm and the logic that must be applied in every ML implementation.

- LogR, neural networks and SVM models knowledge and implementation capacity.

7.3. Future Works

Future works regarding this topic are wide. This chapter will be organized explaining the future works that could follow this bachelor thesis by little modifications to major changes of the assumptions made that could not be done due to time limitations.

For developing a continuation of the work, a computer with better memory capabilities could be used. This will allow the program to be fed by larger dataset and make an analysis of at which data volume will the algorithm be more efficient, evaluate overfitting problems, and generally fixing high variance.

Introducing new dictionaries, not only the one that has been created in this work. Slang and emoji dictionaries, as presented in previous works in section 2.5, as well as polarity dictionaries already created. Allowing to increase the number of features and making the analysis broader.

Study the best parametrization of the fixed values, such as the threshold and *C* parameter. If increased *C* parameter, it could fix situations of high biased and vice versa. Introducing plots of the cost curves and hypothesis and by evaluating those curves, determine the best configurations. The cost is supposed to always decrease and should be compared to the one of the test set.

New data set could be introduced, such as, cross validation dataset, that will make a double comparison of the learning model along with the test set. Therefore, it can improve the performance analysis. Even introducing stochastic gradient descent, for large datasets that have already learned from a training set in order to improve their efficiency.

Apply a ceiling analysis to the overall system, which consists of cheating the system from the beginning step by step, so that it can be seen which step of the process have a higher improvement capacity.

8. REFERENCES

[1] B. Marr, "How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read," Forbes, 21 May 2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#4cb92b9f60ba. [Accessed 1 May 2019].

[2] DOMO, "Data Never Sleeps 6.0," DOMO, [Online]. Available: https://www.domo.com/learn/data-never-sleeps-6. [Accessed 1 May 2019].

[3] J. Neuschimd and T. Vavra, "The next Generation of Data Integration Enabled by Semantic Technologies," IDC Custom Solutions, USA, 2016.

[4] Oxford University Press, "Oxford Dictionaries," [Online]. Available: https://en.oxforddictionaries.com/definition/sarcasm. [Accessed 3 May 2019].

[5] J. Haiman, Talk Is Cheap: Sarcasm, Alienation, and the Evolution of Language, New York: Oxford University Press, 1998.

[6] A. Ng, "Machine Learning, Coursera," Standford University, [Online]. Available: https://www.coursera.org/learn/machine-learning. [Accessed 15 February 2019].

[7] T. M. Mitchell, *The Discipline of Machine Learning,* Pittsburg, PA: Carnegie Mellon university, 2006.

[8] P. Domingos, *A Few Useful Things to Know about Machine Learning,* Seattle, WA: University of Washington, 2012.

[9] N. Donges, "Gradient Descent in a Nutshell," Towards Data Science, 7 March 2018. [Online]. Available: https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0. [Accessed 27 April 2019].

[10] Z. Alissa Almaliki, "Standarization Vs Normalization," Medium, 27 July 2018. [Online]. Available: https://medium.com/@zaidalissa/standardization-vs-normalization-da7a3a308c64. [Accessed 27 April 2019].

[11] S. Swaminathan, "Logistic Regression - Detailed Overview," Towards Data Science, 15º March 2018. [Online]. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc. [Accessed 5 May 2019].

[12] S. S, "Neural Networks," Towards Data Science, 7 April 2018. [Online]. Available: https://towardsdatascience.com/nns-aynk-c34efe37f15a. [Accessed 5 May 2019].

[13] M. Peizeiro, "Introduction to Support Vector Machine," Towards Data Science, 21 January 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-support-vector-machine-svm-4671e2cf3755. [Accessed 20 February 2019].

[14] G. G. Chowdhury, "Natural Language Processing," *Annual review of Information Science and Technology,* vol. 37, no. 1, p. 39, 2003.

[15] S. Raval, "Natural Language Processing and Sentiment Analysis," Medium, 3 February 2017. [Online]. Available: https://medium.com/udacity/natural-language-processing-and-sentiment-analysis-43111c33c27e. [Accessed 15 May 2019].

[16] Z. B. A. a. m. Tejada, "Natural Language Processing," Microsoft Azure, 2 December 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/data-guide/scenarios/natural-language-processing. [Accessed 20 May 2019].

[17] M. Khodak, N. Saunshi and K. Vodrahalli, *A Large Self-Annotated Corpus for Sarcasm*, New Jersey: European Language Resources Association, 2018, p. 6.

[18] R. González-Ibáñez, S. Muresan and N. Wacholder, "Identifying Sarcasm in Twitter: A Closer Look," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: shortpapers,* vol. 1, no. 49, pp. 581-586, 2011.

[19] D. Irazú, C. Bosco, V. Patti and P. Rosso, "Sentiment Polarity Classification of Figurative Language: Exploring the Role of Irony-Aware and Multifaceted Affect Features," *Computational Linguistics and Intelligent Text Processing,* vol. 2, no. 18, pp. 46-57, 2017.

[20] A. G. Prasad, S. S. and S. M. Bhat, "Sentiment Analysis for Sarcasm Detection on Streaming Short Text Data," *IEEE, International Conference on Knowledge Engineering and Applications (ICKEA),* no. 2, pp. 1-5, 2017.

[21] K. Dr H. and M. Barot, "Sarcasm Detection in Sentiment Analysis," *International Journal of Current Engineering and Scientific Research (IJCESR),* vol. 4, no. 9, pp. 24-32, 2017.

[22] D. Jurafsky and J. H. Martin, "N-gram Language Models: Standford University," 23 September 2018. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/3.pdf. [Accessed 23 May 2019].

[23] FreeCodeCamp, "Support Vector Machine," FreeCodeCamp, [Online]. Available: https://guide.freecodecamp.org/machine-learning/support-vector-machine/. [Accessed 15 May 2019].

[24] G. Drakos, "Support Vector Machine vs Logistic Regression," Towards Data Science, 12 August 2018. [Online]. Available:

https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f. [Accessed 15 May 2019].

[25] M. Zhang, Z. Yue and G. Fu, "Tweet Sarcasm Detection Using Deep Neural Network," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, 2016.

[26] Colegios Oficiales de Ingenieros Industriales de Alava, Bizkaia, Gipuzkoa y Navarra, "Ingeniariak," 2017. [Online]. Available: http://www.ingeniariak.eus/wp-content/uploads/2017/04/Encuesta-salarios-Ingenieros-Industriales-2016-2017.pdf. [Accessed 5 June 2019].

[27] T. M. Mitchell, Machine Learning, New York: McGraw-Hill, Inc, 1997.

[28] S. Marsland, Machine Learning, an algorithmic perspective, New Zealand: Chapman & Hall/CRC, 2015.

[29] A. Reyes, P. Rosso and D. Buscaldi, *From Humor Recognition to Irony Detection: The Figurative Language of Social Media,* Mexico DF: Elsevier, 2012.

[30] A. Reyes, P. Rosso and T. Veale, *A Multidimensional Approach for Detecting Irony in Twitter,* Valencia, Sp and Dublin, Ire: Springer Science+Bussiness Media, 2012.

[31] S. Kapadia, "Introduction to Language Models: N-gram," Towards Data Science, 26 March 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9. [Accessed 23 May 2019].